ONEM2M TECHNICAL REPORT

Document Number	TR-0012-V2.0.0
Document Name:	oneM2M End-to-End Security and Group Authentication
Date:	2016-Aug-30
Abstract:	Technical Report of oneM2M End to End Security and Group Authentication

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Contents

Contents 3

1 Scope 8

2 References 8
2.1 Normative references 8
2.2 Informative references 8

3 Definitions, symbols, abbreviations and acronyms 10
3.1 Definitions 10
3.2 Symbols 11
3.3 Abbreviations 11
3.4 Acronyms 11

4 Conventions 11

5 Use Cases 11
5.1 Use Case of End-to-End Authentication in Key Distribution 11
5.1.1 Description 11
5.1.2 Actors 11
5.1.3 Pre-conditions 12
5.1.4 Normal Flow 12
5.1.5 Potential requirements 13
5.2 Use Case of Static Group Authentication (Smart Meter Reading) 13
5.2.1 Description 13
5.2.2 Actors 13
5.2.3 Pre-conditions 13
5.2.4 Normal flow 14
5.2.5 Potential requirements 14
5.3 Use Case of Dynamic Group Authentication (Remote Vehicle Management) 14
5.3.1 Description 14
5.3.2 Actors 14
5.3.3 Pre-conditions 15
5.3.4 Normal Flow 15
5.3.5 Potential requirements 15
5.3.5.1 Static group potential requirements 15
5.3.5.2 Dynamic group potential requirements 15
5.4 Use Case for Secure Group Communication 15
5.4.1 Description 15
5.4.2 Actors 16
5.4.3 Pre-conditions 16
5.4.4 Normal Flow 16
5.4.5 Potential requirements 17
5.5 Use case of End-to-End Authentication 17
5.5.1 Description 17
5.5.2 Actors 17
5.5.3 Pre-Conditions 18
5.5.4 Normal Flow 18
5.5.5 Potential Requirements 18
5.6 Use case of End-to-End Message Authentication using Delegated Means 18
5.6.1 Description 18
5.6.2 Actors 18
5.6.3 Pre-Conditions 19
5.6.4 Normal Flow 19
5.6.5 Potential Requirements 20
5.7 Use case of End-to-End Data Integrity 20
5.7.1 Description 20

5.7.2 Actors 20
5.7.3 Pre-Conditions 21
5.7.4 Normal Flow 21
5.7.5 Potential Requirements 22
5.8 Use case for providing security adaptation at each hop 22
5.8.1 Description 22
5.8.2 Actors 23
5.8.3 Pre-conditions 23
5.8.4 Normal Flow 23
5.8.5 Potential Requirements 24

6 Candidate Architecture 24
6.1 Group Authentication Architecture Proposal 24
6.1.1 Architecture of Static Group Authentication 24
6.1.1.1 Nodes 25
6.1.1.2 Reference Points 25
6.1.2 Group Authentication Requirements 25
6.2 End-to-End Security Framework (ESF) Proposal 1 26
6.2.1 End-to-End Security Framework Introduction 26
6.2.2 ESF Security Layer High Level Architecture 28
6.2.2.1 ESF Security Layer Overview 28
6.2.2.2 ESF Security Layer Requirements 28
6.2.2.2.1 Generic Requirements for the ESF Security Layer 29
6.2.2.2.1.1 Generic ESF Security Layer Macro-Considerations 29
6.2.2.2.1.2 Generic ESF Payload Security Requirements 29
6.2.2.2.1.3 Generic ESF Key Establishment Requirements 29
6.2.2.2.1.4 Generic ESF Facilitation Requirements 30
6.2.2.2.1.5 Generic ESF Envelope Serialization Requirements 30
6.2.2.2.2 ESF-S1 Requirements 31
6.2.2.2.2.1 ESF-S1 Macro-Considerations 31
6.2.2.2.2.2 ESF-S1 Payload Security Requirements 31
6.2.2.2.2.3 ESF-S1 Key Establishment Requirements 32
6.2.2.2.2.4 ESF-S1-Specific ESF Facilitation Requirements 33
6.2.2.2.2.5 ESF-S1 Envelope Serialization Requirements 33
6.2.2.2.3 ESF-Sm Requirements 34
6.2.2.2.3.1 ESF-Sm Macro-Considerations 34
6.2.2.2.3.2 ESF-Sm Payload Security Requirements 34
6.2.2.2.3.3 ESF-Sm Key Establishment Requirements 34
6.2.2.2.3.4 ESF-Sm-Specific ESF Facilitation Requirements 35
6.2.2.2.3.5 ESF-Sm Envelope Requirements 35
6.2.2.3 ESF-S1 Processing flow 36
6.2.2.4 ESF-Sm Processing Flow 39
6.2.3 ESF Preparation Layer and ESF Integration Layer Processing 39
6.2.3.1 ESF Specifications for ESF Target Data Class 1 39
6.2.3.1.1 Profile for ESF Target Data Class 1 39
6.2.3.1.2 ESF Target Data Class 1 Processing at the Sending EEP 40
6.2.3.1.3 ESF Target Data Class 1 Processing at the Receiving EEP 40
6.2.3.2 ESF Specifications for ESF Target Data Class 2 41
6.2.3.2.1 Profile for ESF Target Data Class 2 41
6.2.3.2.2 ESF Target Data Class 2 Processing at the Sending EEP 41
6.2.3.2.3 ESF Target Data Class 2 Processing at the Receiving EEP 42
6.2.3.3 ESF Specifications for ESF Target Data Class 3 43
6.2.3.3.1 Profile for ESF Target Data Class 3 43
6.2.3.3.2 ESF Target Data Class 3 Processing at the Sending EEP 43
6.2.3.3.3 ESF Target Data Class 3 Processing at the Receiving EEP 44

7 Available Options 45
7.1 Review of Existing Technology 45
7.1.1 Review of Object-Based Security Technology 45
7.1.1.1 Introduction to Object-Based Security Technology 45

7.1.1.2 Secure/Multipurpose Internet Mail Extensions (S/MIME) ... 46
7.1.1.2.1 High Level Description of S/MIME ... 46
7.1.1.2.2 Considerations regarding of S/MIME ... 47
7.1.1.2.2.1 CoAP identification of S/MIME media types ... 47
7.1.1.2.2.2 Formatting, Parsing and Canonicalization Complexity for S/MIME ... 47
7.1.1.3 OpenPGP ... 47
7.1.1.3.1 High Level Description of OpenPGP ... 47
7.1.1.3.2 Considerations for OpenPGP ... 48
7.1.1.3.2.1 CoAP identification of the OpenPGP media type ... 48
7.1.1.3.2.2 Formatting, Parsing and Canonicalization Complexity for OpenPGP ... 48
7.1.1.4 XML Security ... 48
7.1.1.4.1 High Level Description of XML Security ... 48
7.1.1.4.2 Considerations for XML Security ... 48
7.1.1.4.2.1 CoAP identification of the XML Security media type ... 48
7.1.1.4.2.2 Formatting, Parsing and Canonicalization Complexity for XML Security ... 49
7.1.1.4.2.3 Canonicalization and XML Security ... 49
7.1.1.5 JSON Security ... 49
7.1.1.5.1 High Level Description of JSON Security ... 49
7.1.1.5.2 Considerations for JSON Security ... 50
7.1.1.5.2.1 CoAP identification of the JSON Security media type ... 50
7.1.1.5.2.2 Formatting, Parsing and Canonicalization Complexity for JSON Security ... 50
7.2 Group Authentication ... 50
7.2.1 Group Authentication Solution 1 ... 50
7.3 A Solution for providing security of data "at-rest" ... 53
7.3.1 General procedure for hosting and accessing secure data ... 53
7.3.2 Bootstrapped procedure for providing data security ... 55
7.3.2.1 Overall Description ... 55
7.3.2.2 Detailed Description ... 55
7.4 A Solution for providing End-to-End Message Authentication using Symmetric Key ... 61
7.4.1 End-to-End Security Credential(s) Generation Process ... 61
7.4.1.1 Overall Description ... 61
7.4.1.2 Detailed Description ... 61
7.5 Proposal for determining detailed Security Requirements, Features and associated Algorithms ... 64
7.5.1 Security Determination Process ... 64
7.5.1.1 Overall Description ... 64
7.5.1.2 Detailed Description ... 64

8 Release 2 End-to-End Security and Rationale ... 67
8.1 Overview of Release 2 End-to-End Security Features ... 67
8.2 Release 2 End-to-End Security of Data (ESData) ... 67
8.2.1 End-to-End Security of Data (ESData) Overview ... 67
8.2.2 End-to-End Security of Data (ESData) Functional Architecture ... 67
8.3 Release 2 End-to-End Security of Primitives (ESPrim) ... 69
8.3.1 End-to-End Security of Primitives (ESPrim) Overview ... 69
8.3.2 End-to-End Security of Primitives (ESPrim) Functional Architecture ... 69
8.4 Release 2 End-to-End Security Certificate-based Key Establishment (ESCertKE) ... 70
8.4.1 End-to-End Security Certificate-based Key Establishment (ESCertKE) Overview ... 70
8.4.2 End-to-End Security Certificate-based Key Establishment (ESCertKE) Functional Architecture ... 70
8.5 Release 2 MAF Security Framework ... 71
8.5.1 MAF Security Framework Overview ... 71
8.5.2 MAF Security Framework Functional Architecture ... 72
8.6 Changes to Release 1 Features in Release 2 ... 73
8.6.1 Changes to Remote Security Provisioning Frameworks (RSPFs) ... 73
8.6.2 Changes to Security Association Establishment Frameworks (SAEFs) ... 74

9 Conclusions and recommendations ... 74

Annex A: Problem Statement for needing End-to-End Data Security ... 74
A.1 Introduction ... 74

Annex B: Use case for remote attestation ... 77

B.1 Description 77
B.2 Actors 78
B.3 Pre-conditions 78
B.4 Normal flow 78
B.5 Potential requirements 79
History 79

1 Scope

The present document provides options and analyses for the security features and mechanisms providing end-to-end security and group authentication for oneM2M.

The scope of this technical report includes use cases, threat analyses, high level architecture, generic requirements, available options, evaluation of options, and detailed procedures for executing end-to-end security and group authentication.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references,only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] oneM2M TR-0004 Definitions and Acronyms

[i.2] W3C Recommendation “Canonical XML Version 1.0”, 2001, http://www.w3.org/TR/xml-c14n

[i.3] IETF RFC 7165: “Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)”

[i.4] IETF RFC 5166: “An Interface and Algorithms for Authenticated Encryption”, 2008

[i.5] oneM2M drafting rules (draft)

[i.6] oneM2M TS-0001 Functional Architecture

[i.7] oneM2M TS-0002 Requirements

[i.8] oneM2M TS-0003 Security Solutions

[i.9] oneM2M TS-0004 Servie Layer Core Protocol Specification

[i.10] W3C Recommendation “XML Signature Syntax and Processing v1.1”, 2013, http://www.w3.org/TR/xmldsig-core1/

[i.11] W3C Recommendation “XML Encryption Syntax and Processing v1.1”, 2013, http://www.w3.org/TR/xmlenc-core1/

[i.12] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[i.13] IETF RFC 6347: "Datagram Transport Layer Security Version 1.2".

[i.14] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings”.

[i.15] IETF RFC 4301: “Security Architecture for the Internet Protocol”, 2005

[i.16] IETF RFC 4880: “OpenPGP Message Format”, 2007

[i.17] IETC RFC 5751: “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification”, 2010

[i.18] Ferguson, Niels & Schneier, Bruce. “Practical Cryptography”. Wiley. p. 333. ISBN 978-0471223573, 2003

[i.19] OASIS Standard, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0”, 2015

[i.20] OASIS Standard, “Common Alerting Protocol Version 1.2”, 2010

[i.21] IETF RFC 7520: “Examples of Protecting Content using JavaScript Object Signing and Encryption (JOSE)”, 2015

[i.22] IETF RFC 2046: “Multipurpose Internet Mail Extensions, (MIME) Part Two: Media Types”, 1996

[i.23] IANA, “Media Types”, http://www.iana.org/assignments/media-types/media-types.xhtml

[i.24] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014.

[i.25] IETC RFC 7252: “The Constrained Application Protocol (CoAP)”, 2014

[i.26] IANA, “Constrained RESTful Environments (CoRE) Parameters, CoAP Content-Formats”: http://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats

[i.27] OASIS Standard “MQTT Version 3.1.1”, 2014, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[i.28] IETC RFC 5652: “Cryptographic Message Syntax (CMS)”, 2009

[i.29] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

[i.30] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1988.

[i.31] IETF RFC 3156: “MIME Security with OpenPGP”, 2001

[i.32] IANA, “Pretty Good Privacy (PGP)”, http://www.iana.org/assignments/pgp-parameters/pgp-parameters.xhtml

[i.33] W3C XML Security Working Group. http://www.w3.org/2008/xmlsec/

[i.34] W3C Recommendation “XML Signature Properties”, 2013, http://www.w3.org/TR/xmldsig-properties/

[i.35] IETF RFC 7518: “JSON Web Algorithms (JWA)”, 2015

[i.36] IETF RFC 7527: “JSON Web Key (JWK)”, 2015

[i.37] IETF RFC 7526: “JSON Web Encryption (JWE)”, 2015

[i.38] IETF RFC 7525: “JSON Web Signature (JWS)”, 2015

[i.39] IETF RFC 4279 "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) "

[i.40] IETF RFC 3629 "UTF-8, a transformation format of ISO 10646"

[i.41] "Unicode Standard Annex #15; Unicode Normalization Forms", Unicode 5.1.0, March 2008. http://www.unicode.org

[i.42] IETF RFC 2014 "HMAC: Keyed-Hashing for Message Authentication"

[i.43] IETF RFC 5869: “HMAC-based Extract-and-Expand Key Derivation Function (HKDF)”, 2010

[i.44] oneM2M TR-0008 Security

[i.45] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory 22 (6): 644–654, 1976

3 Definitions, symbols, abbreviations and acronyms

3.1 Definitions

For the purposes of the present document, the terms and definitions given in [i.1] and the following apply:

M2M Trust Enabler Function (**TEF**): It is a trusted third-party entity that can provide services such as credential generation, registration and provisioning in order to enable secure data protection and access.

End-to-End Security: Provides for securing messages that can traverse multiple hops between communication entities. Securing of messages involves mutually authenticating the end entities. Securing of messages also involves providing confidentiality and integrity protection of messages in order that end entities are assured that the messages have not been altered or eavesdropped by un-authorized entities (including intermediary nodes involved in the transmission).

End-to-End Authentication: Provides an entity with the ability to validate another entity's identity that was supplied as part of the message. The communicating entities can be multiple hops away.

Canonical: A unique and unambiguous representation of data [i.2]

Canonicalization: The process of converting a legal representation of data into its canonical form.

Object-based security: technology that embeds application data within a secure object that can be safely handled by untrusted entities [i.3]

End-to-End Data Integrity Protection: Provides the ability for an entity to integrity protect data.. The integrity protected data can be transported over multiple hops consisting of trusted or untrusted communication entities. An authorized consumer of the data is able to verify the integrity of data and is also able to verify the originator of the data. Such a protection mechanism would ensure that the data is integrity protected "at-rest" and "in-transit" even when handled by intermediate nodes.

End-to-End Data Confidentiality Protection: Provides the ability for an entity to provide for confidentiality protection of data. The confidentiality protected data can be transported over multiple hops consisting of trusted or untrusted communication entities. Only authorized entities can decrypt the confidentiality protected data. Such a protection mechanism would ensure that the data is confidentiality protected "at-rest" and "in-transit" even when handled by intermediate nodes.

Authenticated Encryption with Associated Data: An algorithm providing confidentiality for the plaintext and a way to check its integrity and authenticity while providing the ability to check the integrity and authenticity of some associated data. In this context: plaintext refers to data that is authenticated and encrypted; and associated data refers to data that is authenticated, but not encrypted. See [i.4] for further details.

Group Authentication: Provides an entity (authenticator) with the ability to validate the identities of all entities which belong to a group [i.6]. Confidentiality and integrity of communication between the authenticator and each individual entity in the group is protected from exploit by other entities in the group and any middle node.

NOTE: This may contain additional information.

3.2 Symbols

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

||	Concatenation

3.3 Abbreviations

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

IdAx	Identifier for Entity Ax
IdAy	Identifier for Entity Ay
IdB	Identifier for Entity B
IdC	Identifier for Entity C
Kpsa	Provisioned Credential for M2M Security Association Establishment
KpsaId	Provisioned Credential for M2M Security Association Establishment Identifier
Ks	M2M Group Secure Connection Key
KsId	M2M Group Secure Connection Key Identifier
Rand	Random Number Generated by the Infrastructure Node
MN	Middle Node
IN	Infrastructure Node

3.4 Acronyms

For the purposes of the present document, the abbreviations [i.1] apply:

AMI	Advanced Metering Infrastructure
AEAD	Authenticated Encryption with Associated Data
DAP	Data Aggregation Point

4 Conventions

The key words “Shall”, ”Shall not”, “May”, ”Need not”, “Should”, ”Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.5]

5 Use Cases

5.1 Use Case of End-to-End Authentication in Key Distribution

5.1.1 Description

An oneM2M system may need to transfer sensitive data that should not be exposed to any intermediate nodes or even the application programs in the end nodes, i.e. these data shall only be handled, stored and used in secure environments. One example is to distribute secret keys to the members of a group so that the group members can communicate to each other confidentially. In this case the hop-by-hop security mechanisms cannot meet the required security level, and an end-to-end security mechanism shall be adopted.

The use case in the following sections shows how an end-to-end mechanism could be used to deploy group credentials. For more information about using group credentials seeing Section 5.4.

5.1.2 Actors

The entities involved in this use case are shown in the Figure 5.1.2-1 and described as follows:

M2M Server: It represents an infrastructure equipment that is responsible for creating groups, generating group credentials and transferring group credentials to group members.

M2M Gateway: It represents a gateway that is responsible for forwarding the messages exchanging between M2M Server and target M2M Devices. It also acts as a group agent that is responsible for controlling the entities in the Group-1, Group-2 and Group-3, and broadcasting control commands to these entities.

M2M Device: It represents a device that is responsible for accumulating data from fire sensors, controlling fire doors or fire extinguishing equipments which are attached to this M2M Device.

Group-1: It contains a set of M2M Devices which are responsible for accumulating data from attached fire sensors.

Group-2: It contains a set of M2M Devices which are responsible for controlling attached fire doors.

Group-3: It contains a set of M2M Devices which are responsible for controlling attached fire extinguishing equipments.

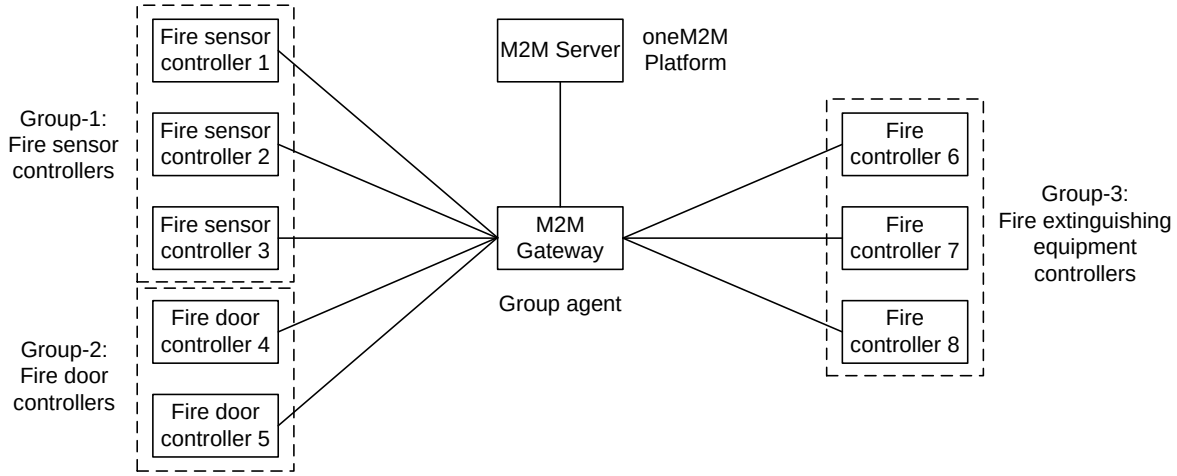


Figure 5.1.2-1: Group credential distribution use case

5.1.3 Pre-conditions

M2M Server, M2M Gateway and M2M Devices are all pre-provisioned with credential(s) that can be used for authentication, data integrity protection and data confidentiality protection.

M2M Devices register to the M2M Gateway in order to communicate with the M2M Server.

5.1.4 Normal Flow

Group credentials distribution procedure:

1. M2M Server creates group resources for the M2M Devices according to their functionality. Group-1 is used for grouping all the M2M Devices that are responsible for accumulating the data from the fire sensors. Group-2 is used for grouping all the M2M Devices that are responsible for controlling the fire doors. Group-3 is used for grouping all the M2M Devices that are responsible for controlling the fire extinguishing equipments.

2. The M2M Server generates group credentials for each group separately.

3. The M2M Server performs an end-to-end authentication with both the M2M Gateway and a target M2M Device with their pre-provisioned credentials. After that a security mechanism used to transfer group credentials is negotiated.

4. The M2M Server encrypts the group credentials using the pre-provisioned credentials shared with the M2M Device and the security method selected in step 3, encapsulates it into a message, and then sends this message to the M2M Gateway.

5. The M2M Gateway forwards the message further to the target M2M Device.

6. The target M2M Device extracts the encrypted content from the message, and then decrypts the encrypted content to get the group credentials.

5.1.5 Potential requirements

1. M2M System shall support end-to-end security providing mutual authentication, security association establishment and remote security provisioning.

2. M2M System shall support establishment of end-to-end security using pre-provisioned credentials.

3. The information exchanged between end entities shall not be exposed to the intermediate nodes.

5.2 Use Case of Static Group Authentication (Smart Meter Reading)

5.2.1 Description

A large number of smart meters are deployed together. The smart meters send meter report frequently through the network to the Utility Data Center / AMI Headend. Furthermore, the Utility Data Center / AMI Headend can issue requests to the smart meters that may be received via an agent (e.g., Data Aggregation Point), e.g., smart meters report, or the Utility Data Center/AMI Headend may need to re-configure all smart meters at the same time.

5.2.2 Actors

The entities involved in this use case are shown in the Figure 5.2.2-1 and described as follows:

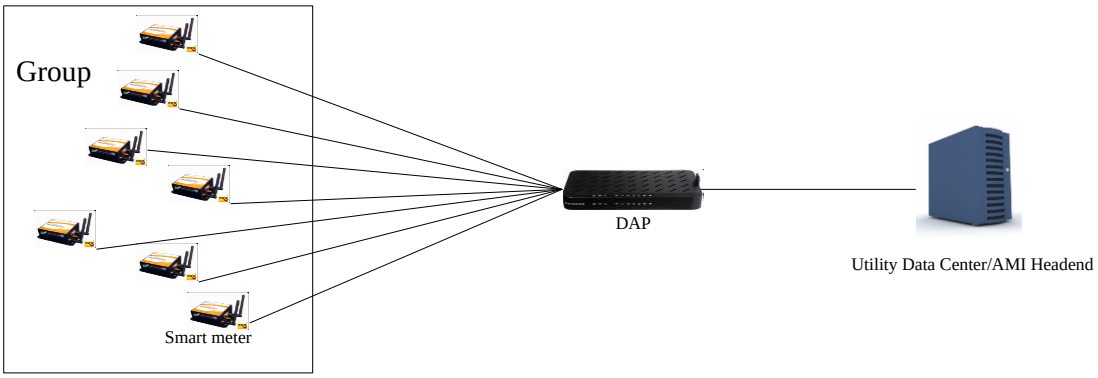


Figure 5.2.2-1: Entities involved in Smart meter reading

Smart meters may form a group of entities that forward meter reports via the group agent to the Utility Data Center/AMI Headend.

DAP (Data Aggregation Point) represents a group agent which acts on behalf of group members to perform mutual authentication with the Utility Data Center/AMI Headend.

Utility Data Center/AMI Headend belongs to the metering service provider and obtains smart meter reports.

5.2.3 Pre-conditions

1. The smart meters are assigned to a group initially.

2. Smart meter, DAP and Utility Data Center/AMI Headend are pre-provisioned with credentials used for performing authentication, respectively.

5.2.4 Normal flow

1. DAP sends a message to indicate that there is a specific group of smart meters that need to communicate with the Utility Data Center / AMI Headend.

2. Each smart meter in the group sends a request via the DAP in order to report to the Utility Data Center / AMI Headend.

3. DAP needs to verify the smart meters identities to upload information before it forwards the request to the Utility Data Center/ AMI Headend.

4. Utility Data Center/ AMI Headend verifies the identity of DAP, and performs mutual authentication with each smart meter in the group simultaneously.

5.2.5 Potential requirements

1. M2M Gateway (i.e., group agent) shall be able to represent M2M Devices (i.e. group members) in order to perform authentication with the M2M Server.

2. The M2M system shall support group authentication to establish security association and enable required procedures for remote provisioning of the M2M Devices (i.e. group members).

5.3 Use Case of Dynamic Group Authentication (Remote Vehicle Management)

5.3.1 Description

Vehicles equipped with communication terminals contain GPS location unit, On-Board Unit, etc. that may send information for purposes such as position tracking, navigation, remote diagnosis, etc., at the same time to the vehicle service centre e.g. to improve vehicle scheduling. Meanwhile, such vehicles may gather in places like airports, train stations, etc. and form groups which can be either static or dynamic, and communicate from vehicles to vehicle service centre via an agent (e.g., remote vehicle gateway or roadside unit).

5.3.2 Actors

The entities involved in this use case are shown in the Figure 5.3.2-1 and described as follows:

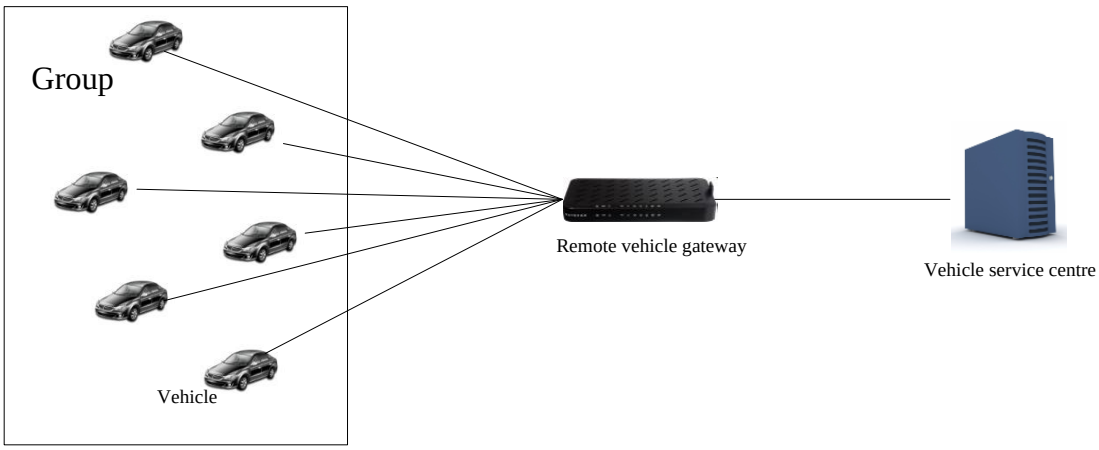


Figure 5.3.2-1: Entities involved in remote vehicle management

Vehicles may form a group of entities that forwards information via the group agent to the vehicle service centre.

The remote vehicle gateway represents a group agent which acts on behalf of group members to perform mutual authentication with the vehicle service centre.

The vehicle service centre belongs to the vehicle service provider and obtains information from the vehicles in a group.

5.3.3 Pre-conditions

1. The vehicles can either be assigned to a group in advance or dynamically changed.
2. The vehicle, remote vehicle and vehicle service centre are pre-provisioned with credentials used for performing authentication, respectively.

5.3.4 Normal Flow

1. The remote vehicle gateway sends a message to indicate that there is a specific group of vehicles that need to communicate with the vehicle service centre.
2. Each vehicle in the group sends request to the remote vehicle gateway at the same time.
3. The remote vehicle gateway verifies the vehicles and then sends request to the vehicle service centre, including vehicle identity list.
4. The vehicle service centre verifies the remote vehicle gateway, and performs mutual authentication with each vehicle on the list simultaneously.

5.3.5 Potential requirements

5.3.5.1 Static group potential requirements

It should statisfy the potential requirements defined in clause 5.2.5 for the static group authentication in remote vehicle management.

5.3.5.2 Dynamic group potential requirements

FFS (For Further Study).

5.4 Use Case for Secure Group Communication

5.4.1 Description

In an oneM2M system the field domain devices may need to be organized into different groups for the purposes of management and operations. For example, in a smart building system the lights, video monitors, air conditioners, ventilation fans, fire sensors, automatic fire extinguishing equipments and fire doors may be managed in various groups. In a group the group members perform the same function. There may be a lot of devices in a group, for the reason of performance and efficiency, the control commands may be broadcasted to the group members instead of contacting them one by one. However, the command issuer is not authenticated by the receiving devices and the commands are not confidentiality and integrity protected. One solution for coping with this issue is using group credentials that are shared among the group members.

The use case in the following sections shows how the group credentials can be used to improve system performance and efficiency.

5.4.2 Actors

The entities involved in this use case are shown in the Figure 5.4.2-1 and described as follows:

M2M Server: It represents an infrastructure node that is responsible for creating groups, generating group credentials and transferring group credentials to group members.

M2M Gateway: It represents a gateway that is responsible for forwarding the messages exchanging between M2M Server and target M2M Devices. It also acts as a group agent that is responsible for controlling the entities in the Group-1, Group-2 and Group-3, and broadcasting control commands to these entities.

M2M Device: It represents a device that is responsible for accumulating data from fire sensors, controlling fire doors or fire extinguishing equipments which are attached to this M2M Device.

Group-1: It contains a set of M2M Devices which are responsible for accumulating data from attached fire sensors.

Group-2: It contains a set of M2M Devices which are responsible for controlling attached fire doors.

Group-3: It contains a set of M2M Devices which are responsible for controlling attached fire extinguishing equipments.

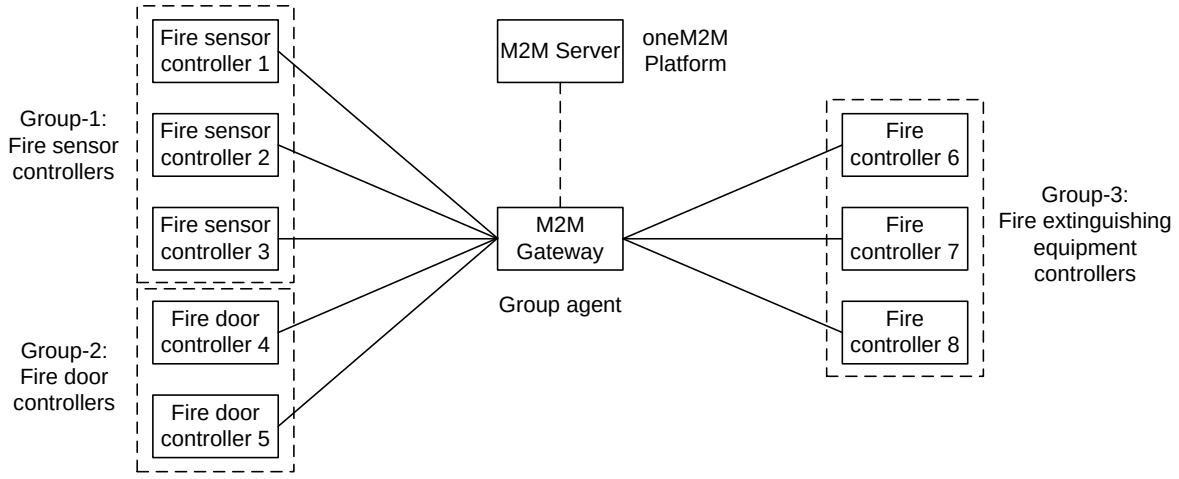


Figure 5.4.2-1: Secure group communication

5.4.3 Pre-conditions

Every M2M Device is provisioned with a group credential that is used for encrypting and/or decrypting group messages.

M2M Gateway is provisioned with all the group credentials that are used for encrypting and/or decrypting group messages.

M2M Devices have the credentials of M2M Gateway in order to authenticate the control commands generated by it.

M2M Devices, M2M Gateway are all issued with a credential that can be used for authentication and data integrity protection..

5.4.4 Normal Flow

Group key using procedure:

1. A fire sensor detects a fire and sends this signal to an M2M Device. The M2M Device generates an alert message that is signed, and then encrypted with the group credential-1. This signed and encrypted message is sent to the M2M Gateway.

2. The M2M Gateway decrypts the encrypted message with the group credential-1 and then verifies the signed message.

3. In case the message is valid, the M2M Gateway generates two control commands. One command asks the M2M Devices in the Group-2 to unlock the fire doors; another command asks the M2M Devices in the Group-3 to switch on the fire extinguishing equipments. All these commands are signed with M2M Gateway's credential. The command for the Group-2 is encrypted with the group credential-2 and the command for the Group-3 is encrypted with the group credential-3. The M2M Gateway then broadcasts these commands.

4. All the M2M Devices belonging to the Group-1, Group-2 and Group-3 can receive these messages. However, only the M2M Devices in the Group-2 can decrypt the message encrypted with the credential-2 and then verify the signed message, and only the M2M Devices in the Group-3 can decrypt the message encrypted with the credential-3 and then verify the signed message.

5. After the received messages are decrypted and verified, the M2M Devices will perform the operations according to the received commands, i.e. the M2M Devices in the Group-2 unlock the fire doors, and the M2M Devices in the Group-3 switch on the fire extinguishing equipments.

5.4.5 Potential requirements

1. A group credential shall be used by a group of members to encrypt/decrypt a broadcast message that is intended for that entire group of members.

2. A security mechanism shall be provided to ensure that a broadcast group message can be authenticated by the group members.

5.5 Use case of End-to-End Authentication

5.5.1 Description

In an oneM2M system, entities may require service layer messaging in order to communicate with another entity that may be multiple hops away. Messages may traverse multiple intermediate entities before the message reaches the final destination (receiver). An entity may require a high-level of assurance that a message originated from a particular end entity. Current hop-by-hop security mechanisms do not meet the requirement and therefore an end-to-end security mechanism shall be adopted.

5.5.2 Actors

The entities involved in the use case are shown in Figure 5.5.2-1 and described as follows:

M2M Application: It represents an application that uses sensor data to perform certain application-specific operations. The M2M Application is multiple hops away from a sensor and may be connected to the sensor by entities that may belong to different administrative domains.

M2M Server: It represents an infrastructure entity that is responsible for enabling an M2M application to obtain services provided by the M2M service provider.

M2M Gateway: It represents a gateway that is responsible for processing and / or forwarding messages that are sent from an M2M Application and from M2M Devices.

M2M Device: It represents a sensor application or a sensor device that is responsible for measuring sensor data and communicating the data to an application.

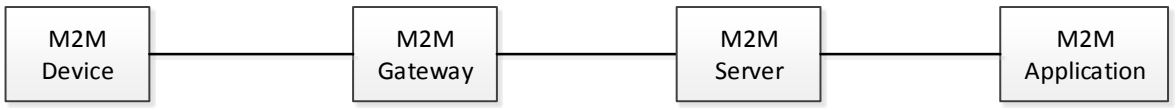


Figure 5.5.2-1: Entities involved in oneM2M messaging

5.5.3 Pre-Conditions

The M2M Application, M2M Gateway, M2M Server and the M2M Device are all provisioned with appropriate credentials that may be used for authentication, message integrity protection, data integrity protection and data confidentiality protection.

M2M Device register with the M2M Gateway in order that an M2M Application may be able to access the sensor data provided by M2M Device.

5.5.4 Normal Flow

Procedure for oneM2M messaging:

1. An M2M Application would like to subscribe to sensor data that is provided by an M2M Device. In order to obtain the data, the M2M Application sends a request message to the M2M Gateway via the M2M Server.
2. The M2M Server receives the message and forwards the message to the appropriate M2M Gateway.
3. The M2M Gateway on receiving the message from the M2M Server, must be able to verify with a high-degree of assurance that the message originated from the M2M Application. In addition, the M2M Gateway may also verify the message has not been tampered or modified by any intermediate entities (e.g. M2M Server). In order to verify the authentication of the message, the M2M Gateway performs an end-to-end authentication of the M2M Application based on the appropriate credentials. If the M2M Gateway is able to positively verify the authentication, the message from the application is processed by the M2M Gateway.

5.5.5 Potential Requirements

1. M2M system shall support establishment of end-to-end security using provisioned end-to-end security credentials.
2. M2M system shall support the capability to perform end-to-end authentication.

5.6 Use case of End-to-End Message Authentication using Delegated Means

5.6.1 Description

In an oneM2M system, entities may require service layer messaging in order to communicate with another entity that may be multiple hops away. Messages may traverse multiple intermediate entities before the message reaches the final destination (receiver). An entity may require a high-level of assurance that a message originated from a particular end entity. A constrained M2M entity would like to delegate certain security functions (e.g. End-to-End Message Authentication) to another trusted entity that may have much more computing power/memory/battery resources.

5.6.2 Actors

The entities involved in the use case are shown in **Figure 5.6.2-1** and described as follows:

M2M Application: It represents an application that uses sensor data to perform certain application-specific operations. The M2M Application is multiple hops away from a sensor and may be connected to the sensor by entities that may belong to different administrative domains.

M2M Server: It represents an infrastructure entity that is responsible for enabling an M2M application to obtain services provided by the M2M service provider.

M2M Gateway: It represents a gateway that is responsible for processing and / or forwarding messages that are sent from an M2M Application and from M2M Devices. The M2M Gateway may function as a Delegated End-to-End Message Authentication Agent on behalf of the M2M Device.

M2M Device: It represents a sensor application or a sensor device that is responsible for measuring sensor data and communicating the data to an application. The M2M Device is assumed to be a very constrained device.

Trusted Third-Party (TTP): It represents an entity that can broker trust relationships between entities that may belong within the same administrative domain or outside. The TTP may also facilitate in providing credential registration as well as credential requisition services.

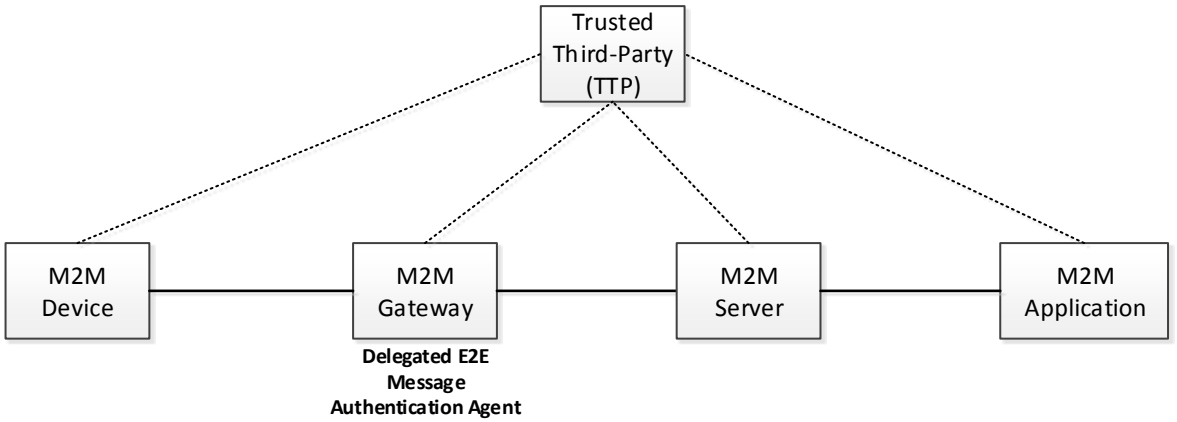


Figure 5.6.2-1: Entities involved in oneM2M messaging

5.6.3 Pre-Conditions

M2M Device registers with the M2M Gateway in order that an M2M Application may be able to access the sensor data provided by M2M Device.

Either by policies determined by M2M service provider or by explicit signalling, the M2M Gateway is assigned as the delegated end-to-end message authentication agent on behalf of the M2M Device

M2M Gateway requests and or registers end-to-end message authentication credentials with a TTP.

Any message originating from the M2M Gateway is integrity protected by the M2M Gateway using end-to-end message authentication credentials on behalf of the M2M Device.

Any message destined to the M2M Gateway or M2M Device is verified to check for authenticity / integrity by the M2M Gateway on behalf of the M2M Device

5.6.4 Normal Flow

Procedure for oneM2M messaging:

4. An M2M Device registers its sensor resource with an M2M Gateway
5. The M2M Gateway based on policies and security requirements determines that the messages originating from it may have to be integrity protected using End-to-End Message Authentication mechanisms using cryptographic mechanisms and therefore may generate appropriate end-to-end credentials, identifiable by a credential identifier and registers the credentials with a TTP. Alternatively, the M2M Gateway may request for appropriate credentials from the TTP, which is then used for protecting the integrity / authenticity of messages originating from it.

6. An M2M Application would like to subscribe to sensor resource that is provided by an M2M Device. In order to obtain the data, the M2M Application sends a request message to the M2M Gateway via the M2M Server.
7. The M2M Server receives the message and forwards the message to the appropriate M2M Gateway.
8. The M2M Gateway on receiving the message from the M2M Server verifies if the M2M Application is authorized to send the request message.
9. Upon verification, the M2M Gateway sends an integrity protected response message to the M2M Application using the end-to-end message authentication credentials that was registered with a TTP via the M2M Server.
10. The M2M Server forwards the integrity protected response message to the M2M Application.
11. Based on the type of credentials (symmetric or assymetric) that has been used for integrity protection of the messagte the M2M Application may or may not be able to verify the integrity / authenticity of the message. If the M2M Application does not have the appropriate credentials, identified by the credential identifier, it requests for the credentials with the TTP.
12. The TTP checks for the authorization of the M2M Application, if authorized, the TTP provisions the M2M Applciation with the appropriate credentials using the credential identifier.
13. The M2M Application is able to verify the integrity / authenticity of the message and processes it.

It must be noted that the mechanisms described above may be used by the M2M Gateway to verify the integrity / authenticity of the request message at Step 5, originating from a M2M Application using similar mechanisms

Message authentication may be performed by verifying a message authentication code / authentication tag that is appended to the original message if using symmetric credentials or by verifying the Digital Signature associated with the message if using asymmetric credentials.

5.6.5 Potential Requirements

1 M2M system shall support an entity to delegate security functions (e.g. message authentication / integrity protection) to a trust-worthy entity on behalf of the originator entity.

5.7 Use case of End-to-End Data Integrity

5.7.1 Description

In an oneM2M system, entities may require service layer messaging in order to communicate data with another entity that may be multiple hops away. Messages that carry data may traverse multiple intermediate entities before the message reaches the final destination (receiver). An entity (receiver) would like to ensure that the data has not been modified by unauthorized entities and to also verify that the authenticity of the data originator (e.g. data producer or data owner or data hosting entity). In addition, an entity that generates, owns or hosts data would like to make sure that only authorized entities are able to view and process the data. The entity in some cases, would like to ensure that even a hosting entity is not able to view and process the data and also ensure that unauthorized intermediate entities are not able to view and process the data.

5.7.2 Actors

The entities involved in the use case are shown in Figure 5.5.2-1 and described as follows:

M2M Application: An application that uses sensor data to perform certain application-specific operations. The M2M Application is multiple hops away from a sensor and may be connected to the sensor by entities that may belong to different administrative domains.

M2M Server: An infrastructure entity that is responsible for enabling an M2M application to obtain services provided by the M2M service provider.

M2M Gateway: A gateway that is responsible for processing and / or forwarding messages that contain sensor data to the M2M Application from M2M Devices. The M2M Gateway may host the data on behalf of the M2M device.

M2M Device: It represents a sensor application or a sensor device that is responsible for measuring sensor data and providing the data that may be hosted on the device or another entity (e.g. M2M Gateway) so that the data is made accessible to an M2M Application.

Trusted Third-Party (TTP): It represents an entity that can broker trust relationships between entities that may belong within the same administrative domain or outside. The TTP may also facilitate in providing credential registration as well as credential requisition services.

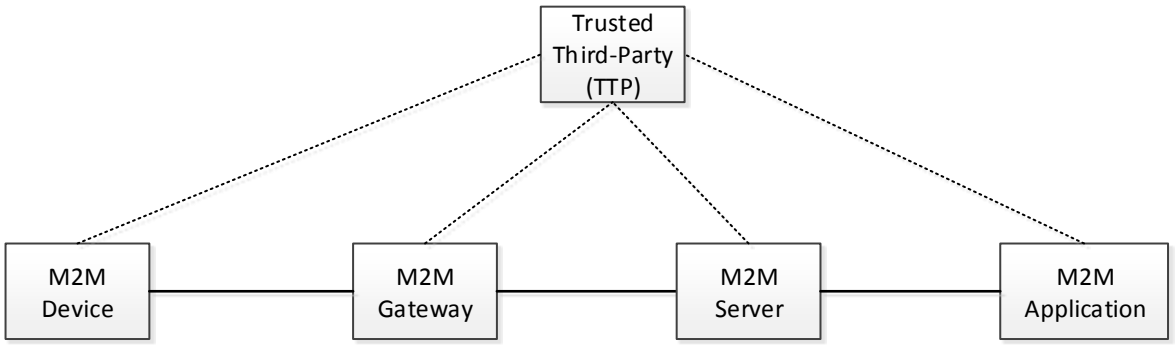


Figure 5.7.2-1: Entities involved in oneM2M messaging

5.7.3 Pre-Conditions

M2M Device registers with the M2M Gateway in order that an M2M Application may be able to access the sensor data provided by M2M Device and optionally hosted on the M2M Gateway or at the M2M Device.

If the M2M Device is hosting the data, then it registers the appropriate end-to-end data integrity credentials used for integrity protection and appropriate end-to-end data confidentiality credentials used for confidentiality protection with the M2M Gateway or with a Trusted Third-Party (TTP).

If an M2M Gateway is hosting the data, then it registers the appropriate end-to-end data integrity credentials used for integrity protection and the appropriate end-to-end data confidentiality credentials used for confidentiality protection with another M2M Gateway or with a TTP

An authorized M2M Application is provisioned with appropriate end-to-end data integrity credentials that is used for verifying the integrity / authenticity of the data and appropriate end-to-end data confidentiality credentials for decrypting the confidentiality protected data.

5.7.4 Normal Flow

Procedure for oneM2M messaging, where the M2M Gateway performs integrity protection and confidentiality protection on behalf of the M2M Device:

1. An M2M Device registers its sensor data with an M2M Gateway
2. The M2M Gateway based on policies and security requirements determines that the data has to be protected for integrity / authenticity and / or confidentiality using cryptographic mechanisms and therefore may generate appropriate credentials, identifiable by credential identifiers and registers the credentials with a TTP. Alternatively, the M2M Gateway may request for appropriate end-to-end data integrity credentials from the TTP, which is then used for protecting the data for integrity / authenticity and appropriate data confidentiality credentials for confidentiality protecting the data. The M2M Gateway hosts the integrity protected and / or confidentiality protected data.

3. An M2M Application would like to subscribe to sensor data that is provided by an M2M Device. In order to obtain the data, the M2M Application sends a request message to the M2M Gateway via the M2M Server.
4. The M2M Server receives the message and forwards the message to the appropriate M2M Gateway.
5. The M2M Gateway on receiving the message from the M2M Server verifies if the M2M Application is authorized to send the request message.
6. Upon verification, the M2M Gateway sends the integrity and confidentiality protected sensor data that is hosted on the M2M Gateway to the M2M Server.
7. The M2M Server forwards the sensor data to the M2M Application.
8. Based on the type of credentials (symmetric or assymetric) that has been used for integrity protection the M2M Application may or may not be able to verify the integrity / authenticity of the sensor data. If the M2M Application does not have the appropriate end-to-end data integrity credentials for verifying the integrity/authenticity of the data, identified by the credential identifier, it requests for the appropriate credentials with the TTP. Similarly, if the M2M Application does have the appropriate end-to-end data confidentiality credentials for decrypting the data, it requests for the appropriate credentials, identitfied by the credential identifier with the TTP.
9. The TTP checks for the authorization of the M2M Application, if authorized, the TTP provisions the M2M Applciation with the appropriate end-to-end data integrity and end-to-end data confidentiality credentials based on the credential identifiers.
10. The M2M Application is able to verify the integrity / authenticity of the sensor data and processes it. The M2M Application is able to decrypt the sensor data.

5.7.5 Potential Requirements

1 The oneM2M system shall enable to protect portions of individual application generated data that is at-rest (e.g. hosted data) for integrity protection and data creator authentication

2 The oneM2M system shall enable to protect portions of individual application data at-rest (e.g. hosted data) for confidentiality protection

3 M2M system shall ensure that the end-to-end data credentials are protected for confidentiality, integrity and against tampering.

4 The M2M system shall ensure that the end-to-end data credentials are protected from exposure to intermediate entities

5.8 Use case for providing security adaptation at each hop

5.8.1 Description

In an oneM2M system, an entity may require service layer messaging in order to communicate data with another entity that may be multiple hops away. Message data may traverse multiple intermediate entities before the message reaches the final destination (receiver). The receiver may require that both the messaging as well as the data are integrity and confidentiality protected for each hop. Because of the diversity of the data and messaging that is generated by each entity belonging to a variety of vertical applications, each application may have varying levels of security requirements. Furthermore, each entity may have varying levels of security capabilities. Therefore, the security capabilities that may be supported at each hop may fall short of, meet or exceed the security requirements of an application. Since the communicating entities may be resource constrained and of differing security capabilities, appropriate security levels have to be selected in order to meet the security requirements of an application on the one hand and match to the security capabilities of the communicating entities on the other hand. A flexible, simple and adaptable security mechanism is therefore desirable for oneM2M where, security associations are established using the appropriate set of security algorithms (e.g. crypto-suites) and associated credentials are supported by two adjacent communicating entities in a hop-by-hop communication session. The security associations may be periodically adjusted based on the security requirements of the specific applications related message communications being supported by communicating entities in an end-to-end manner, for example as sessions are established and torn down.

5.8.2 Actors

The entities involved in the use case are shown in Figure 5.8.2-1 and described as follows:

M2M Application: An application that uses sensor data to perform certain application-specific operations. The M2M Application is multiple hops away from a sensor and may be connected to the sensor by entities that may belong to different administrative domains.

M2M Server: An infrastructure entity that is responsible for enabling an M2M application to obtain services provided by the M2M service provider.

M2M Gateway: A gateway that is responsible for processing and / or forwarding messages that contain sensor data to the M2M Application from M2M Devices. The M2M Gateway may host the data on behalf of the M2M device.

M2M Device: A device that represents a sensor application or a sensor device that is responsible for measuring sensor data and providing the data that may be hosted on the device or another entity (e.g. M2M Gateway) so that the data is made accessible to an M2M Application.

Trusted Third-Party (TTP): Represents an entity that can broker trust relationships between entities that may belong within the same administrative domain or outside. The TTP may also facilitate in providing credential registration as well as credential requisition and provisioning services.

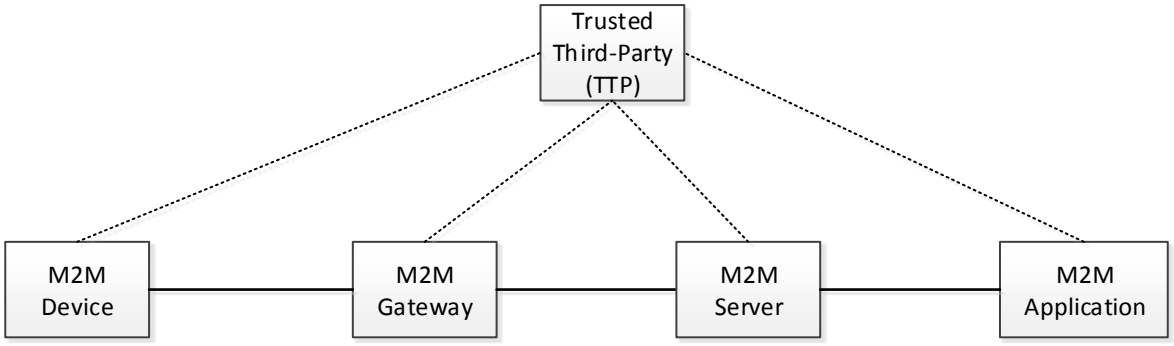


Figure 5.8.2-1: M2M Entities

5.8.3 Pre-conditions

M2M Device registers with the M2M Gateway in order that an M2M Application may be able to access the sensor data provided by M2M Device and optionally hosted on the M2M Gateway or at the M2M Device. The Application on the M2M Device has an associated Security Profile that provides the level of security required in an end-to-end manner.

The M2M Gateway is able to obtain the Security Profile associated with the Application on the M2M Device directly from the M2M Device or from a third-entity.

The M2M entities can request security services (e.g. credential requisition and provisioning) from the TTP.

5.8.4 Normal Flow

Procedure for security adaptation:

1. An M2M Application would like to obtain the sensor data that has been generated by an M2M Device and hosted at an M2M Gateway, perfoms a resource “retrieve” operation request to the M2M Gateway via an M2M Server.

2. The M2M Gateway based on the security profile associated with the M2M Device initiates a security association process with the M2M Server. The cryptosuites and credentials that are selected by the M2M Gateway in order to create the security association will be in accordance to the security requirements as described by the security profile of the M2M Device. The M2M Gateway may negotiate and establish a security association with the M2M Server, or alternatively may use the services of the TTP in order to provision security parameters, for the communications with the M2M Server.
3. Upon creation of the security association, the sensor data is protected according to the security parameters associated with the security association and transported over a secure tunnel (e.g. (D)TLS) from the M2M Gateway to the M2M Server.
4. The M2M Server in conjunction with the M2M Application, creates a separate hop level security association that meets the security requirements associated with the M2M Device in a similar manner to Step 2.
5. The M2M Server transports the sensor data protected over a secure tunnel according to the security parameters associated with the security requirements of the M2M Device.

5.8.5 Potential Requirements

1 The oneM2M system shall enable the ability to provision security profilethat highlights the required security level(s) (e.g. with regards to authentication, integrity and encryption) associated with an entity

2 The oneM2M system shall enable the ability to process security profiles and determine appropriate security features and parameters associated with an entity.

3 The oneM2M sytems shall enable the ability to adapt and establish security associations at each hop of a hop-by-hop communication based on the security requirements associated with the M2M Device.

6 Candidate Architecture

6.1 Group Authentication Architecture Proposal

6.1.1 Architecture of Static Group Authentication

The Candidate Architecture for static group authentication is illustrated as follows:

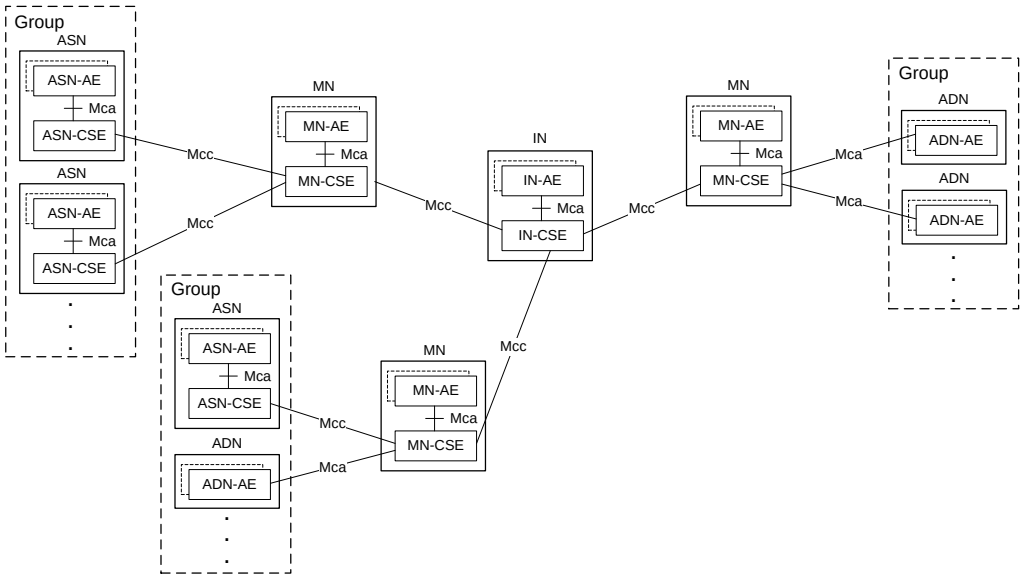


Figure: 6.1.1-1: Illustration of Architecture for Static Group Authentication

6.1.1.1 Nodes

For the architecture of static group authentication in Figure 6.1.1-1, there are three architectural components:

- ASN/ADN [i.6]: an M2M device assigned to a group by the M2M server initially, each M2M device in a given group shall carry out mutual authentication with MN.
- MN [i.6]: an M2M gateway which shall perform mutual authentication with the IN on behalf of all ASNs/ADNs in a given group, it could serve for multiple groups.
- IN [i.6]: an M2M server shall verify the MN's identity and establish the security association with each M2M device in a given group.

NOTE: When the authentication procedure begins, the M2M devices in a given group (e.g. ASN/ADN) will not be changed until the end of this communication session.

6.1.1.2 Reference Points

There are two reference points in the Group Based Authentication Architecture:

- the Mca reference point [i.6] between the ADN-AE and the MN-CSE, or the Mcc reference point [i.6] between the ASN-CSE and the MN-CSE;
- the Mcc reference point [i.6] between the MN-CSE and the IN-CSE.

6.1.2 Group Authentication Requirements

Note: these requirements have been incorporated into TS-0002[i.7].

The following is a list of basic requirements to be considered in design and analysis of group authentication solutions:

- When the M2M Devices are grouped and the M2M Gateway is authorized as delegate of the group for accessing the M2M Server, the M2M Gateway shall be able to, on behalf of the M2M Devices in the group, perform Mutual Authentication with the M2M Server.

- When the M2M Devices are grouped and the M2M Gateway belongs to a third party, oneM2M System shall be able to protect Security and Privacy of communication between individual M2M Device and M2M Server from other M2M devices and the third party M2M Gateway

6.2 End-to-End Security Framework (ESF) Proposal 1

This clause proposes the End-to-End Security Framework (ESF) that provides protection to minimize the number of CSEs that are required to be trusted to maintain the integrity and/or confidentiality of communications in a oneM2M network. ESF includes support for group authentication, which providing mechanisms for improving the efficiency of establishing end-to-end security between members in a group.

The requirements applicable to ESF are provided in SEC-028 through to SEC-033, SEC-036, SEC-046 and SEC-047 of TS-0002 [i.7].

6.2.1 End-to-End Security Framework Introduction

Introduction to the ESF Security Layer. The ESF Security Layer, proposed to be specified in TS-0003 [i.8] defines the following processes:

- Key establishment.
- Transforming *target data* into *ESF-treated target data*, or the reverse, using established keys. The target data is comprised of *plaintext* (data to be encrypted and integrity protected) and/or *additional authenticated data* (to be integrity protected only).
- Serializing (e.g. using JSON or XML) key establishment parameters and/or ESF-treated target data. The serialization is called an *envelope.*

What is ESF? There is no inherent restriction on how the ESF Security Layer can be used – it can be used by entities inside or outside of a oneM2M system. However, the End-to-End Security Framework (ESF) specifies how the ESF Security Layer is used when one or both of the ESF Security Layer end-points is a CSE or AE. These additional processing details depend on the *target data class* to be secured, listed in Table 6.2.1-1 "List of ESF target data classes". The target data class specific details are proposed to be specified in TS-0001 [i.6].

NOTE 1: The list of supported ESF target data classes is limited for this release. Further ESF target data classes could be added in the future releases.

Input Item	Target data			Details in Clause:
	Plaintext	Additional Authenticated Data (AAD)	Class Identifier	
Request or response	Entire Request or response	None	1	6.2.3.1
Request or response	***Content*** parameter	None	2	6.2.3.2
<*contentInstance*> resource	*content* attribute	None	3	6.2.3.3

Table 6.2.1-1 List of ESF target data classes

NOTE: The currently-defined ESF target data classes do not use the Additional Authenticated Data. The option to use Additional Authenticated Data has been included in the ESF Security Layer so that future ESF target data classes can use Additional Authenticated Data without requiring an update to the ESF Security Layer specifications.

The ESF functionality within a CSE or AE is called an *ESF End-Point (EEP)*. The ESF reference model partitions the functionality into three protocol layers; shown in Figure 6.2.1-1 "High Level ESF Reference model".

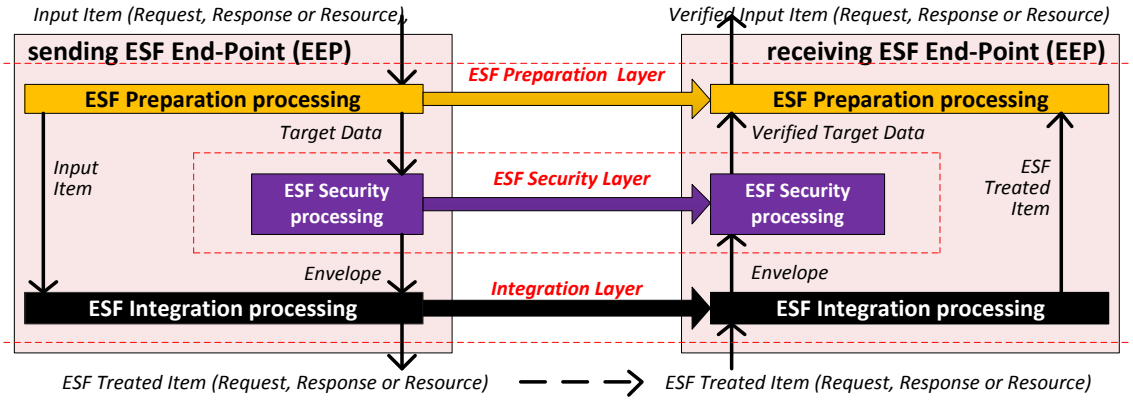


Figure 6.2.1-1 High Level ESF Reference model. Control data (e.g. security parameters) are not shown.

The three protocol layers in the high level ESF reference model are:

- **ESF Preparation Layer**:
    - A sending EEP, given an *input item*, extracts the appropriate target data. The input items are either oneM2M messages or oneM2M resources. The target data is provided to the ESF security layer, in a format independent of the target data class, for transforming into an envelope. The input item is forwarded to the ESF Integration Layer for composing the ESF-treated item.
    - A receiving EEP reconstitutes a verified input item from the verified target data received from the ESF Security Layer and the ESF-treated item and ESF target data identifier received from the sending EEP.

  The processing at this layer depends on the ESF target data class.

- **ESF Security Layer**: transforms the target data into an envelope, or the reverse. The processing at this layer does not depend on the target data class. Further details are in clause 6.2.2 "ESF Security Layer High Level Architecture".

- **ESF Integration Layer:**
    - A sending EEP forms the ESF-treated item from the input item received from the ESF Preparation Layer, and the envelope received from the ESF Security Layer.
    - A receiving EEP obtains the ESF-treated item via the Mcc and/or Mca reference points, and extracts the envelope. The envelope is provided to the ESF security layer for transforming back to the verified target data. The ESF-treated item is provided to the ESF Preparation layer for reconstituting the verified input item.

  The processing at this layer depends on the ESF target data class.

The ESF Preparation Layer and ESF Integration Layer are tightly dependent on each other, since they both depend on ESF target data class, however the ESF Security Layer processing is independent of those layers. Aside from the dependence of key generation on the ESF target data identifier, the security processing is independent of the ESF target data. This independence allows a single ESF Security Layer end-point implementation to be used for all ESF target data classes. Three security session types are proposed for the ESF Security Layer (for more details, see clause 6.2.2 "ESF Security Layer High Level Architecture"). Each of the supported ESF target data classes can be protected using any of the ESF security session types.

In which oneM2M specifications are the ESF layers proposed to be detailed? The functional architecture of the ESF Preparation Layer and ESF Integration Layer are proposed to be specified in TS-0001 [i.6], with protocol-level details proposed to be specified TS-0004 [i.9]. As stated at the beginning of the present clause, the ESF Security Layer details are proposed to be specified in TS-0003 [i.8].

6.2.2 ESF Security Layer High Level Architecture

6.2.2.1 ESF Security Layer Overview

Three security session types, listed in Table 6.2.2.1-1, have been proposed for the Security Layer.

Term	Abbr	Security Session Comprises	State Lifetime	Details in clause:
One-way Single envelope security session	ESF-S1	One envelope sent in one direction. Provides all crypto parameters necessary to establish the keys securing the payload. Other than credentials, no state is stored	Stateless	6.2.2.3
Two-way Single Envelope security session	ESF-S2	Envelopes securing request and corresponding response. Request envelope provides all crypto parameters necessary to establish the key plus secured payload. Other than credentials, no state is stored	Initiator stores state until response received. Target stores state until. response sent	None. See note.
Two-way Multi-envelope security session	ESF-Sm	Handshake exchange followed by any number of envelopes containing secured payloads (sent in either direction)	Initiator and Target store state until session is terminated or expires	6.2.2.4
NOTE: The ESF-S2 security session type is not presently considered a priority, and was not examined in further detail. The ESF-S2 security session type is mentioned here for future reference.				

Table 6.2.2.1-1 List of security session types in the ESF reference model.

The ESF-S1 and ESF-Sm security session types were seen as priorities for oneM2M Release 2. Consequently, the ESF-S2 session type was not examined in further detail when the present report was composed. The remaining discussion of the ESF Security Layer focusses only on the ESF-S1 and ESF-Sm security session types.

For each security session type the ESF Security Layer processing can be partitioned into:

- *Key Establishment* processing: includes
    - negotiating the cryptographic algorithms to be used for key establishment and payload security, and
    - establishing keys for securing payloads and. KEF
- *Payload Security* processing: applying security mechanisms transforming the target data into the ESF-treated target data, or the reverse, using the established keys.
- *Envelope Serialization* processing: responsible for representing exchanged data (key establishment parameters, and ESF-treated target data) in an envelope: a common serialization independent of the security session type.

6.2.2.2 ESF Security Layer Requirements

This clause proposes detailed requirements for the ESF Security Layer. These requirements are intended only for use in design and analysis of components of ESF as part of the work represented in this TR.

Clause 6.2.2.2.1 proposes requirements which are independent of the security session type. Clauses 6.2.2.2.2 and 6.2.2.2.3 proposes requirements specific to the security session types ESF-S1 and ESF-Sm respectively.

6.2.2.2.1 Generic Requirements for the ESF Security Layer

6.2.2.2.1.1 Generic ESF Security Layer Macro-Considerations

"Macro-considerations" are considerations or requirements for the ESF Security Layer that are not specific to the Facilitation, Key Establishment, Payload Security or Envelope Serialization processing of the ESF Security Layer.

The following is a list of proposed macro-considerations for the ESF Security Layer which are independent of security session type:

- The ESF Security Layer supports ESF End-Point implementations with ESF Key Establishment processing in a secure environment providing established keys to ESF Payload Security processing outside the secure environment.

Proposed macro-considerations for individual security session types are found in clauses 6.2.2.2.2.1 (for ESF-S1), and 6.2.2.2.3.1 (for ESF-Sm).

6.2.2.2.1.2 Generic ESF Payload Security Requirements

The following is a list of proposed requirements for the ESF Payload Security processing which are independent of the security session type:

- ESF Payload Security supports protecting the payload using a symmetric (secret) key established using the corresponding ESF Key Establishment reference points.
- ESF Payload Security supports encryption of the input payload using a symmetric key.
- ESF Payload Security supports integrity protection of the input payload using a symmetric key.
- ESF Payload Security supports a range of crypto algorithms, to enable migrating to another secure cryptographic algorithms in the event that cryptographic algorithms are found to be insecure or are otherwise deprecated.
- ESF Payload Security is not required to maintain state between security sessions.

Proposed ESF Payload Security requirements for individual security session types are found in clauses 6.2.2.2.2.2 (for ESF-S1), and 6.2.2.2.3.2 (for ESF-Sm).

6.2.2.2.1.3 Generic ESF Key Establishment Requirements

The following is a list of proposed requirements for the ESF Key Establishment processing which are independent of the security session type:

- ESF Key Establishment provides a mechanism for negotiating an appropriate set of cryptographic algorithms and credentials. This could be influenced by
    - Cryptographic algorithms supported by the EEPs
    - Credentials of the EEPs.
    - A Security Profile relevant to the target data.
- ESF Key Establishment supports authenticated key establishment based on a symmetric (secret) key provisioned by means unspecified by oneM2M specifications.
- ESF Key Establishment supports authenticated key establishment based on a symmetric (secret) key obtained through interaction with a trusted Facilitator.

NOTE: The role of Facilitator in this case could be played by an M2M Authentication Function (MAF) or M2M Enrolment Function (MEF).

- ESF Key Establishment supports key establishment using certificates.

- ESF Key Establishment supports mutual authentication.
- ESF Key Establishment could support one-way authentication of the source/initiator EEP or target/responder EEP.
- ESF Key Establishment ensures that there is only a very low probability that keys established for one security session are identical to keys established for a different security session – regardless of the security session type of both security sessions.
- ESF Key Establishment ensures keys established for one security session type cannot be used in another security session type; e.g. the keys established for a single-envelope security session cannot be used for a multi-envelope security session.
- ESF Key Establishment ensures that there is only a very low probability that keys established for one ESF target data class are identical to keys established for a different ESF target data class.
- ESF Key Establishment maintains minimal state between security sessions.

Proposed ESF Key Establishment requirements for individual security session types are found in clauses 6.2.2.2.2.3 (for ESF-S1), and 6.2.2.2.3.3 (for ESF-Sm).

6.2.2.2.1.4 Generic ESF Facilitation Requirements

ESF Facilitation is independent of the security session type. The following is a list of proposed requirements for the ESF Facilitation which are independent of the security session type:

- ESF Facilitation is independent of the security session type (S1, S2 or Sm)
- ESF Facilitation supports being secured from the ESF End-Point to the Facilitator.
- ESF Facilitation supports mutual authentication of the ESF End-Point and the Facilitator.
- A Facilitator may apply access control.
- ESF Facilitation supports EEPs obtaining symmetric (secret) keys through interaction with trusted Facilitators.

NOTE: The role of Facilitator in this case could be played by an M2M Authentication Function (MAF) or M2M Enrolment Function (MEF).

- ESF Facilitation supports Facilitators acting as a repository of certificates or public keys associated with an ESF End-Point.
- ESF Facilitation supports Facilitators acting as a repository of Security Profiles.

Proposed ESF Facilitation requirements imposed by individual security session types are found in clauses 6.2.2.2.2.4 (for ESF-S1), and 6.2.2.2.3.4 (for ESF-Sm).

6.2.2.2.1.5 Generic ESF Envelope Serialization Requirements

The following is a list of proposed ESF Envelope Serialization requirements which are independent of the security session type:

- The envelope serialization is independent of the security session type (S1, S2 or Sm) and type of input data (resource portion or primitive).
- The envelope serialization identifies the security session type (S1, S2 or Sm)
- The envelope serialization identifies the target data class.
- The envelope serialization is compatible with oneM2M resource and primitive representations (JSON and XML are supported at the time of writing).

- The envelope serialization shall allow simple identification of envelopes within oneM2M resources representations (JSON and XML are supported at the time of writing).

Proposed ESF Envelope Serialization requirements for individual security session types are found in clauses 6.2.2.2.2.5 (for ESF-S1), and 6.2.2.2.3.5 (for ESF-Sm).

6.2.2.2.2 ESF-S1 Requirements

6.2.2.2.2.1 ESF-S1 Macro-Considerations

There are no ESF-S1-specific macro-considerations in addition to the macro-considerations in clause 6.2.2.2.1.1 "ESF Security Layer Macro-Considerations", the following ESF-S1-specific macro-considerations are proposed:

6.2.2.2.2.2 ESF-S1 Payload Security Requirements

A Payload Security Algorithm Class (psAlgCl) describes the type of protection that is afforded the input payload; suggested Payload Security Algorithm Classes for oneM2M Release 2 are listed in Table 6.2.2.2.2.2-1 "List of ESF-S1 Payload Security Algorithm Class (AlgCl) options". This list is not necessarily exhaustive, and other options could be available.

The selection of algorithms for each AlgCl is discussed in clause 7 "Available Options".

Payload Security Algorithm Class	Example mechanism	Brief Description	Target Data	Keys that must be established
Symmetric integrity protection only	MIC	Source EEP and Target EEP(s) use a symmetric key for integrity protection, but not confidentiality. Target EEPs can impersonate the Source EEP. For this psAlgCl, ESF does not perform encryption.	AAD only	Payload security processing requires a symmetric envelope master key established between the Source EEP and Target EEP(s)
Symmetric confidentiality and integrity protection	AEAD	Source EEP and Target EEP(s) use a symmetric key for confidentiality and integrity protection. Target EEPs can impersonate the Source EEP	Plaintext + (optional) AAD	
Symmetric confidentiality and non-repudiation protection	AEAD + Digital Signature	Source EEP and Target EEP(s) use a symmetric key for confidentiality and digital signature for non-repudiation. Target EEPs cannot impersonate the Source EEP. Use of Authenticated Encryption is suggested.	Plaintext + (optional) AAD	Payload security processing requires a symmetric envelope master key established between the Source EEP and Target EEP(s); additionally the Source EEP uses a digital signature private signing key and the Target EEP(s) use a digital signature public verification key.
Non-repudiation protection only	Digital Signature	Source EEP and Target EEP(s) use a digital signature for non-repudiation. For this psAlgCl, ESF does not perform encryption.	AAD only	Source EEP uses a digital signature private signing key and the Target EEP(s) use a digital signature public verification key.

Table 6.2.2.2.**2.2-1 List of ESF-S1 Payload Security Algorithm Class (psAlgCl) options.**

6.2.2.2.2.3 ESF-S1 Key Establishment Requirements

The psAlgCl in Table 6.2.2.2.2.2-1 "List of ESF-S1 Payload Security Algorithm Class (AlgCl) options" which require a digital signature public verification key are:

- AEAD + Digital Signature, and
- Digital Signature.

If one of these psAlgCl are used, then a digital signature public verification key is provided for additional verification of the input payload. The digital signature public verification key could be provided in a certificate communicated as key establishment parameter in envelope or via Facilitator. The Target EEP verifies the certificate prior to verifying the digital signature on the target data.

The psAlgCl in Table 6.2.2.2.2.2-1 "List of ESF-S1 Payload Security Algorithm Class (psAlgCl) options" which require an envelope master key are:

- Symmetric integrity protection
- Symmetric confidentiality and integrity protection, and
- Symmetric confidentiality and integrity protection + non-repudiation.

A Key Establishment Algorithm Class (keAlgCl) describes the type of key establishment mechanisms used to establish an envelope master key; suggested Key Establishment Algorithm Classes for oneM2M Release 2 are listed in Table 6.2.2.2.2.3-1 "List of ESF-S1 Key Establishment Key Establishment Algorithm Class (AlgCl) options". This list is not necessarily exhaustive, and other options could be available. This list is not intended to constrain the options considered for establishing an envelope master key.

Name	Brief Description
Trusted Third Party (TTP)	The Source EEP requests a Facilitator to store the envelope master key with an associated set of permissions. The Facilitator could optionally generate the envelope master key for the Source EEP. The Source EEP and Facilitator agree on a unique envelope master key identifier for the envelope master key, and this envelope master key identifier is provided in the envelope. After the Target EEP retrieves the envelope, then the Target EEP provides the envelope master key identifier to the Facilitator. If the Target EEP has the necessary permissions, then the Facilitator provides the envelope master key to the Target EEP. This requires a high degree of trust in the Facilitator, and communication with the Facilitator (from both the Source EEP and Target EEP) must be mutually authenticated and secured.
Pre-Provisioned Shared Key	The envelope master key is provided to the Source EEP and Target EEP(s) via mechanisms not specified by oneM2M (e.g. manual input, pre-provisioning).
Key Encryption	The Source EEP selects a set of Target EEPs for which the Source EEP knows a public encryption key of the Target EEP (for example, the public encryption keys could be in certificates, or Identity Based Encryption might be applied) or the Source EEP shares a symmetric key encryption key (KEK). For each Target EEP, then Source EEP encrypts the envelope master key using either the public encryption key or KEK and includes the resulting encrypted envelope master key in the envelope. The envelope includes an encrypted envelope master key for each Target EEP. After the Target EEP retrieves the envelope, then the Target EEP uses its private decryption key or KEK (shared with the Source EEP) to decrypt its encrypted envelope master key and obtain the envelope master key. The mechanism could optionally authenticate the Source EEP; e.g by including a digital signature that can be verified using a certificate of the Source EEP.
Key Agreement	(*This approach can work only if there is a single Target EEP*). The Source EEP and Target EEP applies a key agreement protocol to establish a shared key. For example, the Source EEP could use a Diffie-Hellman protocol [i.45] using a public key in a certificate of the Target EEP. The mechanism could optionally authenticate the Source EEP; e.g. by including a digital signature that can be verified using a certificate of the Source EEP.

Table 6.2.2.2**.2.3-1 List of ESF-S1 Key Establishment Algorithm Class (AlgCl) options which could be considered for establishing an envelope master key.**

6.2.2.2.2.4 ESF-S1-Specific ESF Facilitation Requirements

Proposed ESF-S1-specific Requirements. In addition to the generic requirements in clause 6.2.2.2.1.3 “Generic ESF Key Establishment Requirements”, the following ESF-S1-specific requirements are proposed:

1. ESF Facilitation shall support Facilitators acting as a repository of long-term key parameters associated with an ESF End-Point which could include
    - List of supported ksAlgSets.
    - List of supported psAlgSets.
    - List of Certificates or public keys.

   These parameters could be used in the Key Encryption, Key Agreement and Digital Signature Algorithm Classes in Table 6.2.2.2.2.3-1 “List of ESF-S1 Key Establishment Algorithm Class (AlgCl) options”

2. ESF Facilitation may support Facilitator acting as a repository of the latest provided short-term key parameters associated with an ESF End-Point, which could include
    - A short-lifetime randomly-generated, non-secret value – e.g. for use as challenges in authentication or for adding non-secret entropy to key generation.
    - A short-lifetime public key value – e.g. EC-DH public key for adding perfect forward secrecy (PFS).

NOTE: The case of using a short-lifetime secret value – e.g. for use as adding secret entropy to key generation – is already covered by the first ESF Facilitation requirement “The reference point supports EEPs obtaining symmetric (secret) keys through interaction with trusted Facilitators” in clause 6.2.2.2.1.3 “Generic ESF Key Establishment Requirements”.

These parameters could be used in the Key Encryption, Key Agreement and Digital Signature Algorithm Classes in Table 6.2.2.2.2.3-1 “List of ESF-S1 Key Establishment Algorithm Class (AlgCl) options”

6.2.2.2.2.5 ESF-S1 Envelope Serialization Requirements

Proposed ESF-S1 Envelope Serialization requirements. In addition to the generic requirements in clause 6.2.2.2.1.5 “Generic ESF Envelope Serialization Requirements”, the following ESF-S1-specific Envelope Serialization requirements are proposed:

- The ESF-S1 Envelope Serialization supports representing secured payloads for the Payload Security Algorithm Classes in clause 6.2.2.2.2.2 “ESF-S1 Payload Security Requirements”.
- The ESF-S1 Envelope Serialization supports representing key establishment parameters for the Key Establishment Algorithm Classes in clause 6.2.2.2.2.3 “ESF-S1 Payload Security Requirements”.
- The ESF-S1 Envelope Serialization supports representing the keAlgSet and psAlgSet”.

JSON Representations. The IETF JOSE specifications [i.3] can provide a JSON representation of the ESF-S1 secured payload and key establishment parameters. The generic envelope serialization requirements and the proposed ESF-S1 Envelope requirements could be satisfied by an envelope which is a JSON element comprised of

- An identifier for the security session type (in this case indicating S1), followed by
- One or more JOSE data elements containing the ESF-S1 key establishment parameters and ESF-S1 secured payload.

XML Representations. The W3C XML-SIG [i.10] and XML-ENC [i.11] can provide a XML representation of the ESF-S1 secured payload and key establishment parameters. The generic envelope serialization requirements and the proposed ESF-S1 Envelope requirements could be satisfied by an envelope which is a XML element comprised of

- An identifier for the security session type (in this case indicating S1), followed by

- One or more XML-SIG and/or XML-ENC data elements containing the ESF-S1 secured payload and key establishment parameters.

6.2.2.2.3 ESF-Sm Requirements

6.2.2.2.3.1 ESF-Sm Macro-Considerations

In addition to the generic considerations in clause 6.2.2.2.1.1 “ESF Security Layer Macro-Considerations”, the following ESF-S1-specific macro-considerations: are proposed

- The ESF-Sm security layer shall support a signaling messages including the ability to end an existing ESF-Sm session. An example of such a signaling messages are the alert messages of TLS v1.2 (see RFC 5246 [i.12]) and DTLS v1.2 (see RFC 6347 [i.13]).

6.2.2.2.3.2 ESF-Sm Payload Security Requirements

Proposed ESF-Sm Payload Security Requirements. In addition to the generic requirements in clause 6.2.2.2.1.2 “Generic Payload Security Requirements”, the following ESF-Sm-specific Payload Security requirements are proposed:

- Replay detection shall be supported by ESF-Sm.

Possible ESF-Sm Payload Security Solutions.

Encryption and Integrity protection can be provided by either

- A combination of an encryption algorithm and an independent MIC algorithm.
- An Authenticated Encryption with Associated Data (AEAD) algorithm.

Replay detection can be provided by including a sequence number which is protected by the integrity protection calculation.

All of the above mechanisms are supported by appropriate choices of ciphersuites for TLS v1.2 in RFC 5246 [i.12] and DTLS v1.2 in RFC 6347 [i.13]. The selection of appropriate ciphersuites is discussed in clause 7 “Available Options”.

TLS assumes a reliable transport (such as TCP); consequently, TLS cannot be used in all oneM2M scenarios. DTLS does not assume a reliable transport; consequently, DTLS can be used in all oneM2M scenarios. For this reason, DTLS is preferable to TLS.

There are other security protocols which could provide the functionality required for ESF-Sm Payload Security, however these are not as widely used as TLS and DTLS.

Recommendation: DTLS v1.2 record payload protection is the recommended solution for ESF-Sm Payload Security.

6.2.2.2.3.3 ESF-Sm Key Establishment Requirements

Proposed ESF-Sm Key Establishment Requirements. In addition to the generic requirements in clause 6.2.2.2.1.3 “Generic ESF Key Establishment Requirements”, the following ESF-Sm-specific Key Establishment requirements are proposed:

- Perfect Forward Secrecy countermeasures (e.g. the ephemeral Diffie-Hellman protocol or ephemeral Elliptic Curve Diffie-Hellman protocol) is required to be supported by ESF-Sm and optional to be used.
- Replay detection of ESF-Sm Key Establishment handshake messages is required to be supported by ESF-Sm Key Establishment and required to be used.

This list is not necessarily exhaustive, and other options could be available. This list is not intended to constrain the options considered for establishing symmetric (secret) keys for use in the EF-Sm Payload Security reference point.

Possible ESF-Sm Key Establishment Solutions. All of the above mechanisms are supported by appropriate choices of ciphersuites for TLS v1.2 (see RFC 5246 [i.12]) and DTLS v1.2 (see RFC 6347 [i.13]).

TLS assumes a reliable transport (such as TCP); consequently, TLS cannot be used in all oneM2M scenarios. DTLS does not assume a reliable transport; consequently, DTLS can be used in all oneM2M scenarios. For this reason, DTLS is preferable to TLS.

There are other security protocols which could provide the functionality required for ESF-Sm Payload Security, however these are not as widely used as TLS and DTLS.

Recommendation: The DTLS v1.2 handshake is the recommended solution for ESF-Sm key establishment.

6.2.2.2.3.4 ESF-Sm-Specific ESF Facilitation Requirements

There are no ESF-Sm-specific requirements proposed in addition to the generic requirements in clause 6.2.2.2.1.3 "Generic ESF Facilitation Requirements".

NOTE: In theory, ESF-Sm could support a Facilitator enabling an faster ESF-S1 key establishment by acting as a repository of the latest provided short-term, non-secret key establishment parameters associated with an ESF End-Point. Such short-term parameters could include

- A short-lifetime secret value – e.g. for use as adding secret entropy to key generation. This options requires trusting the Facilitator to maintain the confidentiality of the value.
- A short-lifetime public key value – e.g. EC-DH public key for adding perfect forward secrecy (PFS).

There is a problem, because the DTLS v1.2 handshake (recommended for ESF-Sm Key Establishment handshake) does not support integrating such parameters into a faster key exchange. If DTLS v1.2 handshake is used, then storing short-term, non-secret key establishment parameters for ESF-Sm Key Establishment provides no value. For this reason, there is no proposal to store short-term, non-secret key establishment parameters for ESF-Sm key establishment.

6.2.2.2.3.5 ESF-Sm Envelope Requirements

Solutions providing ESF-Sm Key Establishment and Payload Security. The discussion in previous clauses indicates the following:

- The DTLS v1.2 handshake (see RFC 6347 [i.13]) is the preferred existing solution providing the functionality required for ESF-Sm Key Establishment key establishment (see clause 6.2.2.2.3.3).
- The DTLS v1.2 record payload protection is the preferred existing solution providing the functionality required for ESF-Sm Payload Security payload security (see clause 6.2.2.2.3.3).

Consequently, DTLS v1.2 is a preferable solution for ESF-Sm Key Establishment and ESF-Sm Payload Security.

However, DTLS messages are binary data, while resources and primitives use a JSON or XML representation. This prevents using the binary DTLS messages directly in a resource or primitive

Proposed ESF-Sm Envelope requirements. In addition to the generic requirements in clause 6.2.2.2.1.5 "Generic ESF Envelope Requirements", the following ESF-Sm-specific Envelope requirements are proposed:

- ESF-Sm Envelope is required to support transporting DTLS records, where these records contain
  - DTLS handshake messages, providing the ESF-Sm Key Establishment functionality
  - DTLS protected payloads, providing the ESF-Sm Payload Security functionality. Since DTLS allows tunneling DTLS handshake messages inside an established DTLS session, the DTLS protected payloads can provide the ESF-Sm Key Establishment functionality.
- ESF-Sm Envelope is required to encode binary DTLS records in an ASCII character space allowed by the values in JSON and XML data elements.

JSON Representations. The generic envelope serialization requirements and the proposed ESF-Sm Envelope requirements could be satisfied by an envelope which is a JSON element comprised of

- An identifier for the security session type (in this case indicating Sm), and
- A JSON data element containing a base64 encoding [i.14] of a DTLS message.

XML Representations. The generic envelope serialization requirements and the proposed ESF-Sm Envelope requirements could be satisfied by an envelope which is a XML element comprised of

- An identifier for the security session type (in this case indicating Sm), and
- A XML data element containing a base64 encoding [i.14] of a DTLS message.

6.2.2.3 ESF-S1 Processing flow

ESF-S1 supports an envelope having a single Source EEP and one or more Target EEPs.

Figure 6.2.2.3-1 shows the processing flow when using this session type.

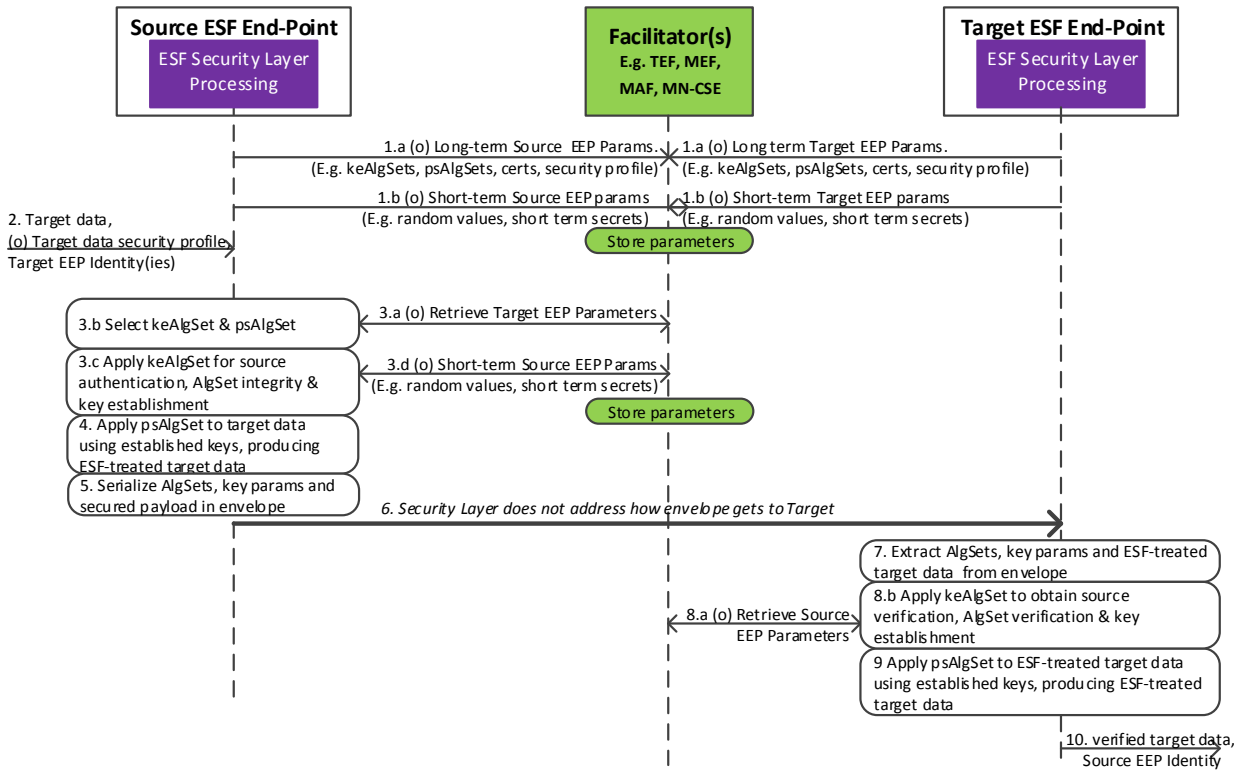


Figure 6.2.2.3-1 Processing flow for single envelope session type (S1).

1. EEPs can interact with Facilitators to provide the Facilitators with long-term parameters (that is, parameters with long lifetime) and/or short-term parameters (that is, parameters with a short lifetime). This step is not expected to be mandatory.

    a. (Optional). EEPs could provide interaction with one or more Facilitators to provide long-term parameters (that is, parameters with long lifetime) that the other EEPs can use either for

        - Generating S1 session envelopes, as Source EEPs, or

- Processing received envelopes, as Target EEPs.

Example long-term parameters include

- List of supported Key Establishment Algorithm-Sets (keAlgSets)
- List of supported Payload Security Algorithm-Sets (psAlgSets)
- Certificates
- Security Profile

This interaction is expected to happen relatively infrequently – possibly only once in the lifetime of the EEP. The Facilitator(s) store the parameters and make them available for retrieval. Access control rules could be provided to the Facilitator(s) so that Facilitator(s) may restrict access to the parameters.

b. (Optional). EEPs could provide interaction with one or more Facilitators to update short-term parameters that can be provided to other EEPs either for

- Generating S1 session envelopes, as Source EEPs, or
- Processing received envelopes, as Target EEPs.

Example, short-term parameters include

- Random values with short lifetime; these might be used as challenges for authentication.
- Secret keys with short lifetime; these might be used as challenges for authentication.

The Facilitator(s) store the parameters and make them available for retrieval.

This interactions are expected to happen relatively infrequently – possibly only once in the lifetime of the EEP. Access control rules could be provided to the Facilitator(s) so that the Facilitator(s) may restrict access to the parameters.

2. The Source EEP provides the Security Layer Functions with

- Target Data, and
- (Optional) Security profile applicable to the Target Data,
- Identifiers for the intended Target EEPs. This could include individual identifiers and group identifiers.

This triggers the resulting call flow.

3. **Source EEP Key Establishment processing**:

NOTE: The present document does not specify how the Source EEP determines the minimum security levels to be applied and what session type should be applied, although this could be considered in future versions. Interaction with security profiles should be considered. It is also worth considering where the decision is made – ESF Preparation Layer, ESF Security Layer or ESF Integration Layer.

a. (Optional) The Source EEP could retrieve the Target EEP(s)' long-term and/or short-term parameters from the Facilitator(s). Access to the parameters may be controlled, for example, using <*accessControlPolicy*> resources. The Facilitator(s) could be a CSE(s) with which the Target EEP is registered, although other options are also possible.

b. The Source EEP selects one or more Key Establishment Algorithm-Sets (keAlgSet) and Payload Security Algorithm-Set (psAlgSet).

NOTE: Multiple keAlgSets could be needed only in some (but not all) scenarios involving multiple Target EEPs.

The selection of Algorithm Sets could be influenced by

- The combinations of keAlgSets and psAlgSets supported by the Source EEP.
- Preferences configured to the Source EEP .
- The combinations of keAlgSets and psAlgSets supported by the Target EEP(s), which may have been either configured to the Source EEP or retrieved by the Source EEP from the Facilitators in (optional) step 3.a.

The choices of keAlgSet(s) and psAlgSet are somewhat dependent on each other; in particular, if the psAlgSet includes encryption and/or symmetric-key based integrity protection of the target data, then the keAlgSet(s) must include an algorithm for generating a secret key to be used for this protection. A high level overview of possible psAlgSet Options is found in clause 6.2.2.2.2.2 "ESF-S1 Payload Security Requirements". A high level overview of possible keAlgSet options is found in clause 6.2.2.2.2.3 "ESF-S1 Key Establishment Requirements".

c. The Source EEP applies the selected keAlgSet(s), which can include
   - Proof of the Source EEP's identity,
   - Proof of the integrity of the selected keAlgSet(s) and psAlgSet, and
   - Generation of keys for processing the target data.

d. (Optional) The Source EEP provides the Facilitator(s) with updated Short-term Key establishment parameters. This may include configuring access controls at the Facilitator to limit access only to the set of intended Target EEPs.

4. **Source EEP Payload Security processing**. The Source EEP applies the selected psAlgSet to the input payload using the key(s) provided by the Source EEP, resulting in the ESF-treated target data.
5. **Source EEP Envelope Serialization**: The Source EF serializes the keAlgSet(s), psAlgSet, key establishment parameters and ESF-treated target data into an envelope.
6. The envelope is obtained by the Target EEP(s). The Security Layer does not address how the Target EEP(s) obtain the envelope.

NOTE: The following discussion shows processing at one of the potentially multiple Target EEPs.

7. **Target EEP Envelope Serialization processing:** The Target EF extracts the keAlgSet(s), psAlgSet, key establishment parameters and ESF-treated target data from the received envelope.
8. **Target EEP Key Establishment processing**: In the case that there are multiple Key Establishment Algorithm Sets present, the Target EEP identifies a Key Establishment Algorithm Set that it expects to process successfully.

   a. (Optional) The Target EEP could retrieve the Source EEP's long-term and/or short-term parameters from the Facilitator(s). The Facilitator(s) could authorize access based on configured access controls.

   b. The Target EEP applies the selected Key Establishment Algorithm Set, which can include
      - Verifying the identity of Source EEP,
      - Verifying the selected keAlgSet(s) and psAlgSet, and
      - Generation of keys for processing the input payload.

9. **Target EEP Payload Security processing** : The Target EEP applies the selected psAlgSet to the secured payload using the established key(s), resulting in the verified target data.
10. The Target EEP outputs:
    - The verified target data, and
    - Source EEP's identity.

6.2.2.4 ESF-Sm Processing Flow

The ESF-Sm reference model allows establishing multi-envelope sessions between a single Initiating EEP (analogous to a TLS/DTLS Client) and a single Terminating EEP (analogous to a TLS/DTLS Server).

An ESF-SM security session has two phases:

- **ESF-Sm handshake phase**: in which the Initiating EEP and Terminating EEP – optionally assisted by Facilitator(s), and under the influence of a security profile – accomplish the following:
    - Negotiating a key establishment cryptographic algorithms and payload security cryptographic algorithms.
    - Apply the negotiated key establishment cryptographic algorithms to
        - Authenticate the Terminating EEP,
        - (Optionally) Authenticate the Initiating EEP,
        - Establish secret session keys to be used in for payload security.

  A processing flow would depend on the handshake details, and the handshake details are beyond the scope of a high level description. For this reason, a processing flow is not provided.

- **ESF-Sm payload security phase**, in which the sending EEP (which can be the Initiating EEP and Terminating EEP) apply the negotiated payload security cryptographic algorithms transform the target data into ESF-treated target data which is then serialized in an envelope. At the receiving EEP, the ESF-treated target data is extracted from the envelope and the negotiated payload security cryptographic algorithms transform the ESF-treated target data into verified target data. Due to the simplicity of this processing, a processing flow diagram is not provided.

6.2.3 ESF Preparation Layer and ESF Integration Layer Processing

6.2.3.1 ESF Specifications for ESF Target Data Class 1

6.2.3.1.1 Profile for ESF Target Data Class 1

Table 6.2.3.1.1-1 "Profile for ESF Target Data Class 1" contains the profile for ESF Target Data Class 1.

Aspect	Description
ESF Target Data Class Identifier	1
Input item	A request or response
Target data	A serialization of the entire request or response
Treated Item	A new encapsulating request or response whose ***Content*** parameter is the envelope produced by the ESF Security Layer. The Originator and Hosting CSE of the encapsulating message are the ESF End-Points for this message.
EEP Constraints	The present document allows an EEP to be the Originator or Hosting CSE of the input message or any other CSE on the oneM2M path between the Originator and Hosting CSE of the input message.
Security Layer	There can be only one Target EEP.

Constraints	
Target Data-class specific parameters for ESF Policy	Parameters for the encapsulating message (e.g. ***To***, ***From***)

Table 6.2.3.1.1-1 Profile for ESF Target Data Class 1

6.2.3.1.2 ESF Target Data Class 1 Processing at the Sending EEP

The following steps occur at a Sending EEP when applying this ESF Target Data Class to an outbound request or response

1. The EEP's Preparation Function serializes the request or response into a binary data object.
2. The EEP's Preparation Function passes to the EEPs Security Function
    a. The applicable ESF Target Data Class identifier.
    b. Target data set to the binary data object and
    c. There is no additional data for this ESF Target Data Class.
3. The EEPs Security Function applies the applicable processing according to the security session type, generating an envelope.

    NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two outbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

4. The EEPs Security Function provides the envelope and applicable ESF Target Data Class identifier to the EEP's Integration Function. The envelope is serialized as is XML or JSON.
5. The EEP's Integration Function forms a request or response (according to if the protected message was a request or response) including the envelope in the content:
6. The EEP's Integration Function passes the request or response to the next step in processing outbound messages.

6.2.3.1.3 ESF Target Data Class 1 Processing at the Receiving EEP

The following steps occur at a Receiving EEP when applying this ESF Target Data Class to an inbound request or response

1. The EEP's Integration Layer parses the received message to identify the sending EEP and extract the envelope which is the ***Content*** parameter of the message. The envelope is serialized as is XML or JSON.
2. The EEP's Integration Function passes to the EEPs Security Function
    a. The applicable ESF Target Data Class identifier.
    b. Envelope.
    c. There is no additional data for this ESF Target Data Class.
3. The EEPs Security Function applies the applicable processing according to the security session type, generating a verified target data. This processing includes verification that the provided ESF Target Data Class identifier is correct.

NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two inbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

4. The EEPs Security Function provides the verified target data and to the EEP's Preparation Layer.
5. The EEP's Preparation Function parses the binary data object to form the verified request or response.
6. The EEP's Preparation Function passes the request or response to the next step in processing inbound messages.

6.2.3.2 ESF Specifications for ESF Target Data Class 2

6.2.3.2.1 Profile for ESF Target Data Class 2

Table 6.2.3.2.1-1 "Profile for ESF Target Data Class 2" contains the profile for ESF Target Data Class 2.

Aspect	Description
ESF Target Data Class Identifier	2
Input item	A request or response
Target data	A serialization of the ***Content*** parameter of the input request or response
Treated Item	The input request or response with the original ***Content*** parameter replaced by the envelope produced by the ESF Security Layer.
EEP Constraints	The present document allows an EEP to be the Originator or Hosting CSE of the input request or response or any other CSE on the oneM2M path between the Originator and Hosting CSE of the input request or response.
Security Layer Constraints	There can be only one receiving EEP.

Table 6.2.3.2.1-1 Profile for ESF Target Data Class 2

6.2.3.2.2 ESF Target Data Class 2 Processing at the Sending EEP

The following steps occur at a Sending EEP when applying this ESF Target Data Class to an outbound request or response

7. The EEP's Preparation Function extracts the ***Content*** parameter of the request or response, and serializes the ***Content*** parameter to form the target data.
8. The EEP's Preparation Function passes to the EEPs Security Function
    - The applicable ESF Target Data Class identifier.
    - Target data, and
    - There is no additional data for this ESF Target Data Class.
9. The EEPs Security Function applies the applicable processing according to the security session type, generating an envelope.

    NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two outbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

10. The EEPs Security Function provides the envelope and applicable ESF Target Data Class identifier to the EEP's Integration Function. The envelope is serialized as is XML or JSON.

11. The EEP's Integration Function forms the ESF-treated request or response from the input request or response by updating values of the following parameters

    - ***Content:*** The original value is replaced with the envelope received from the ESF Security Function.
    - ***Resource Type:*** replacing this with a reserved identifier for ESF-protected ***Content*** parameter.

12. The EEP's Integration Function passes the ESF-treated request or response to the next step in processing outbound messages.

6.2.3.2.3 ESF Target Data Class 2 Processing at the Receiving EEP

The following steps occur at a Receiving EEP when applying this ESF Target Data Class to an inbound request or response

7. The EEP's Integration Layer parses the received message to identify the sending EEP and extract

    - ***Content***: comprises the envelope. The envelope is serialized as is XML or JSON.
    - ***Resource Type:*** a reserved identifier for protected requests and responses protected by ESF Target Data Class 2.

8. The EEP's Integration Function passes to the EEPs Security Function

    - The applicable ESF Target Data Class identifier.
    - Envelope.
    - There is no additional data for this ESF Target Data Class.

9. The EEPs Security Function applies the applicable processing according to the security session type, generating a verified target data. This processing includes verification that the provided ESF Target Data Class identifier is correct.

NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two inbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

10. The EEPs Security Function provides the EEP's Preparation Layer with

    - Verified target data, comprising a serialization of the original value of the ***Content*** parameter.
    - There is no additional data for this ESF Target Data Class

11. The EEP's Preparation Function

    a. Parses the target data to form the value of the verified ***Content*** parameter.

    b. (If the ***Content*** parameter contains a resource) Determines the resource type of the verified ***Content*** parameter.

    c. Forms the verified request or response by updating values of the following parameters

        - ***Content:*** The value in the received request or response is replaced with the value received from the ESF Security Function in the target data.
        - (if the verified ***Content*** parameter contains a resource) ***Resource Type:*** replacing this with the resource type of the resource in the verified ***Content*** parameter.

12. The EEP's Preparation Function passes the verified request or response to the next step in processing inbound messages.

6.2.3.3 ESF Specifications for ESF Target Data Class 3

6.2.3.3.1 Profile for ESF Target Data Class 3

Table 6.2.3.3.1-1 "Profile for ESF Target Data Class 3" contains the profile for ESF Target Data Class 3.

Aspect	Description
ESF Target Data Class Identifier	3
Input item	A <*contentInstance*> resource
Target data	A serialization of • The *content* attribute in the input <*contentInstance*> resource. • (Optionally) A *contentInfo* attribute if it is applicable to the input <*contentInstance*> resource, regardless of whether the *contentInfo* attribute is explicitly included in input <*contentInstance*> resource, or whether *ontologyRef* is inherited from the parent of the input <*contentInstance*> resource. • (Optionally) An *ontologyRef* attribute applicable to the input <*contentInstance*> resource, regardless of whether the *ontologyRef* attribute is explicitly included in input <*contentInstance*> resource, or whether *ontologyRef* is inherited from the parent of the input <*contentInstance*> resource.
Treated Item	The input <*contentInstance*> resource, with the original *content* attribute value replaced by the envelope produced by the ESF Security Layer.
EEP Constraints	The EEPS could be any entities on the data path of the *content* attribute value - including CSEs, AEs or an entities in another system with which oneM2M interworks.
Security Layer Constraints	Multiple Target EEPs are allowed.

Table 6.2.3.3.1-1 Profile for ESF Target Data Class 3

6.2.3.3.2 ESF Target Data Class 3 Processing at the Sending EEP

The following steps occur at a Sending EEP when applying this ESF Target Data Class to an outbound <*contentInstance*> resource.

13. The EEP's Preparation Function processes the outbound <*contentInstance*>
    a. The EEP's Preparation Function parses the outbound <*contentInstance*> the extract the attributes to be protected, which comprises:
        - The *content* attribute.
        - (Optionally) The *contentInfo* attribute.
        - (Optionally) The *ontologyRef* attribute.
    b. The EEP's Preparation Function serializes these attributes to form the target data.
14. The EEP's Preparation Function passes to the EEPs Security Function
    - The applicable ESF Target Data Class identifier.
    - Target data.
    - There is no additional data for this ESF Target Data Class.

15. The EEPs Security Function applies the applicable processing according to the security session type, generating an ESF envelope.

    NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two outbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

16. The EEPs Security Function provides the envelope and applicable ESF Target Data Class identifier to the EEP's Integration Function. The ESF envelope is serialized as in XML or JSON.

17. The EEP's Integration Function forms the ESF-treated <*contentInstance*> from the input <*contentInstance*> by updating values of the following attributes

    - *content:* The original value is replaced with the ESF envelope received from the ESF Security Function.

18. The EEP's Integration Function passes the ESF-treated resource to the next step in processing outbound resources.

6.2.3.3.3 ESF Target Data Class 3 Processing at the Receiving EEP

The following steps occur at a Receiving EEP when applying this ESF Target Data Class to an inbound request or response

13. The EEP's Integration Layer parses the received message to identify the sending EEP and extract

    - *content:* comprising the ESF envelope received from the ESF Security Function.

14. The EEP's Integration Function passes to the EEPs Security Function

    - The applicable ESF Target Data Class identifier.
    - Envelope.
    - There is no additional data for this ESF Target Data Class.

15. The EEPs Security Function applies the applicable processing according to the security session type, generating a verified target data. This processing includes verification that the provided ESF Target Data Class identifier is correct.

NOTE: If ESF-Sm is applied, and the security session is not yet established, then the ESF-Sm handshake phase must be applied prior to transmitting any target data. Consequently, the first two inbound envelopes in this case will typically contain handshake phase key exchange parameters, and not the message.

16. The EEPs Security Function provides the EEP's Preparation Layer with

    - Verified target data, comprising a serialization of the original value of the *content*, (optionally) *contentInfo* and (Optionally) *ontologyRef*.
    - There is no additional data for this ESF Target Data Class

17. The EEP's Preparation Function

    a. Parses the verified target data to form the verified values of *content*, (optionally) *contentInfo* and (Optionally) *ontologyRef*.

    b. Forms the verified resource by updating values of the following attributes

        - *Content:* The value in the received resource is replaced with the value of *content* received from the ESF Security Function in the target data.

- *contentInfo*: If the verified target data contains *contentInfo*, then this value replaces the value received in the verified target data.
- *ontologyRef*: If the verified target data contains *ontologyRef*, then this value replaces the value received in the verified target data.

18. The EEP's Preparation Function passes the verified resource to the next step in processing inbound resources.

7 Available Options

7.1 Review of Existing Technology

7.1.1 Review of Object-Based Security Technology

7.1.1.1 Introduction to Object-Based Security Technology

NOTE: this clause borrows heavily from the introduction to [i.3].

Channel-based security technologies such as IPsec [i.15], Transport Layer Security (TLS) [i.12] and Datagram TLS (DTLS) [i.13] create a secure channel at the IP layer or transport layer over which data can flow. In protocols with application-layer intermediaries, channel-based security protocols would protect messages from attackers between intermediaries, but not from the intermediaries themselves.

In the present oneM2M security specifications, the only protection afforded by oneM2M messages is provided by the channel-based TLS or DTLS. Some of the use cases in clause 5 require protecting messages from CSEs acting as intermediaries, and the existing oneM2M security mechanisms cannot offer this protection. Additional technology is required.

Object-based security technologies (see definitions) embed data within "secure object" that can be safely handled by untrusted intermediaries in the scenarios discussed above. Clause 7.1.1 provides a review of well-known standardized object-based security technologies including

- **Secure/Multipurpose Internet Mail Extensions (S/MIME**) [i.17] was proposed as a mechanism for providing an email with end-to-end security protection in the presence of untrusted Mail Transfer Agents en route to its destination. S/MIME provides confidentiality, integrity, and data origin authentication. S/MIME assumes a hierarchical PKI. These specifications are discussed in clause 7.1.1.2.
- **OpenPGP** [i.16] provides security services similar to S/MIME. Open PGP supports both hierarchical PKIs (as used in S/MIME) and a decentralized PKI known as a "Web-of-Trust" [i.18]. OpenPGP is discussed in clause 7.1.1.3.
- The XML security specifications **XML Signature** [i.10] and **XML Encryption** [i.11] can be applied to any content type, with the result represented in an XML object. These mechanisms are used by several security token systems (e.g., Security Assertion Markup Language (SAML) [i.19), and the Common Alerting Protocol (CAP) emergency alerting format [i.20]. XML security discussed in clause 7.1.1.4.
- The IETF **JSON Object Signing and Encryption (JOSE)** working group [i.21] has been chartered to develop a secure object format based on JSON with roughly equivalent features to the XML security specifications in the preceding bullet. JSON Security is discussed in clause 7.1.1.5.

In addition to a high level description of the protocols, the following issues are considered:

oneM2M Protocol Binding support for natively identifying the object-based security protocol media type. Internet media types [i.22] are identified by a string, such as "application/xml", registered in the IANA Media Types

registry [i.23]. All the above object-based security protocols use one of the registered media types. Note that the media types currently supported by oneM2M systems are defined in clause 6.7 of TS-0004 [i.9].

- HTTP [i.24]: HTTP natively supports identification of all registered media string-based identifiers, including the media type for the above object-based security protocols.
- CoAP [i.25]: CoAP In order to minimize the overhead of using the string-based media type identifiers, CoAP natively recognizes only the small set of Internet media types recorded in the "CoAP Content-Formats" sub-registry of the IANA "CoRE Parameters" registry (see [i.26]). Recognition of the object-based security protocol media type in CoAP is examined on a case by case basis.
- MQTT [i.27]: MQTT leaves identification of the media type to the application layer. For all choices of object-based security protocol, oneM2M would be required to specify how the object-based security protocol media type is identified when MQTT is used.

Formatting and Parsing Complexity: some of the above object-based security protocols include complex rules for formatting and parsing of messages. This is worth consideration, because application developers that lack the tools or motivation to handle complex rules are likely to avoid developing applications using such security protocols. Execution environments could provide function calls that apply complex formatting and/or parsing on behalf of AEs– thus reducing the burden on the application developers. However, it is unclear if the scope of oneM2M includes defining function calls. Consequently, it is unclear if the formatting and parsing complexity is a factor in making decisions. This review (clause 7.1.1) only reports on the formatting and parsing complexity; the review avoids drawing any recommendations based on the complexity.

Canonicalization: Consider a scenario where a signed object of some media type (e.g. XML) is parsed at an intermediary server, with the information later reconstructed same media type before being forwarded to another entity. For many media types, there are multiple legitimate equivalent serializations (representations) of the original signed object, so the second serialization of the object may differ slightly from the original serialization in the object. If the serializations differ, then a signature on the original serialization of the object would no longer apply for the second serialization– even though the serializations are logically equivalent. Consequently, an entity who receives the second serialization of the object cannot use the original signature to verify the origin of the object.

For example, in XML the nature of the whitespace may not convey any meaning – so the intermediary server may reconstruct XML with different whitespace to the original serialization. Similarly, the order of attributes in XML does not convey any meaning, so the intermediary server may reconstruct XML with attributes in a different order to the original serialization. In both cases, a signature on the original serialization of the object would no longer apply for the second serialization– even though the serializations are logically equivalent.

To address this issue, some media types define a *canonical form* that is uniquely and unambiguously representable in the environment where the signature is created and the environment where the signature will be verified. The process of producing the canonical form from a particular serialization is called *canonicalization*.

Canonicalization has the advantage that signatures can always be verified, even if the object gets parsed and reconstructed by an intermediary. This benefit does not come for free, since canonicalization can add to the complexity of formatting and parsing. Furthermore, if the original serialization of the object is always sent with the digital signature, then the complexity of canonicalization provides no technical benefit. Consequently, canonicalization can be an advantage in some scenarios and disadvantage in others.

In summary: canonicalization is required in scenarios where the object is parsed and reconstructed at an intermediate server. However, if objects are not parsed and reconstructed then canonicalization simply adds an un-necessary and complex step. This review (clause 7.1.1) only reports whether the object-based security technologies provide canonicalization; the review does not investigate which scenarios warrant canonicalization, and avoids drawing any recommendations based on the support for canonicalization.

7.1.1.2 Secure/Multipurpose Internet Mail Extensions (S/MIME)

7.1.1.2.1 High Level Description of S/MIME

NOTE: This description borrows heavily from the S/MIME specification [i.17].

S/MIME [i.17] (Secure/Multipurpose Internet Mail Extensions) provides a consistent way to send and receive secure MIME data. S/MIME provides authentication, message integrity and non-repudiation of origin (using digital signatures), and data confidentiality (using encryption). As a supplementary service, S/MIME provides for message compression.

S/MIME is not restricted to mail; it can be used with any transport mechanism that transports MIME data, such as HTTP or SIP. As such, S/MIME takes advantage of the object-based features of MIME and allows secure messages to be exchanged in mixed-transport systems.

S/MIME defines the creation and processing of a MIME body part that has been cryptographically enhanced according to the Cryptographic Message Syntax (CMS) [i.28]. CMS is used to digitally sign, digest, authenticate, or encrypt arbitrary message content. The CMS values are generated using ASN.1 [i.29], using BER-encoding (Basic Encoding Rules) [i.30].

7.1.1.2.2 Considerations regarding of S/MIME

7.1.1.2.2.1 CoAP identification of S/MIME media types

S/MIME [i.17] registers the internet media type identifiers "`application/pkcs7-mime`" and "`application/pkcs7-signature`" in the IANA Media Types registry [i.23].

At the time of writing, the S/MIME media types are not in the "CoAP Content-Formats" registry [i.26], so CoAP cannot natively identify the S/MIME media types. If oneM2M decides to use S/MIME, then oneM2M will need to specify how the CoAP binding indicates that S/MIME has been used.

7.1.1.2.2.2 Formatting, Parsing and Canonicalization Complexity for S/MIME

Recall that S/MIME uses CMS which in turn uses ASN.1 [i.29], with BER-encoding (Basic Encoding Rules) [i.30].

[i.3] states the following opinion regarding the use of ASN.1

> "In recent years, usage of ASN.1 has decreased (along with other binary encodings for general objects), while more applications have come to rely on text-based formats such as the Extensible Markup Language (XML) [W3C.REC-xml] or the JavaScript Object Notation (JSON) [RFC7159].
>
> "Many current applications thus have much more robust support for processing objects in these text-based formats than ASN.1 objects; indeed, many lack the ability to process ASN.1 objects at all."

S/MIME provides simple guidance for canonicalization of text. Otherwise, S/MIME does not impose any canonicalization rules, but requires that the original MIME entity is already canonicalized according to the media type and subtype of the original MIME entity. The complexity of this canonicalization then depends on the media type of the original MIME entity.

For example, if S/MIME is used to secure XML, then XML canonicalization must be applied (see clause 7.1.1.4.2.3). XML Security also requires canonicalization, so the overhead of canonicalization is the same where S/MIME or XML security is applied.

7.1.1.3 OpenPGP

7.1.1.3.1 High Level Description of OpenPGP

The OpenPGP message format specification [i.16] uses a combination of strong public-key and symmetric cryptography to provide security services for electronic communications and data storage. These services include confidentiality, key management, authentication, and digital signatures.

OpenPGP supports both hierarchical PKIs (as used with S/MIME) and a decentralized PKI known as a "Web-of-Trust". The web-of-trust model is mostly useful for authenticating people, but has not gained significant momentum.

7.1.1.3.2 Considerations for OpenPGP

7.1.1.3.2.1 CoAP identification of the OpenPGP media type

MIME Security with OpenPGP [i.31] defines three content types in the IANA Media Types registry [i.23] for implementing security and privacy with OpenPGP: "`application/pgp-encrypted`", "`application/pgp-signature`" and "`application/pgp-keys`".

At the time of writing, the OpenPGP media types are not in the "CoAP Content-Formats" registry [i.26], so CoAP cannot natively identify the OpenPGP media types. If oneM2M decides to use OpenPGP, then oneM2M will need to specify how the CoAP binding indicates that OpenPGP has been used.

7.1.1.3.2.2 Formatting, Parsing and Canonicalization Complexity for OpenPGP

OpenPGP messages are ASCII radix-64 representations of binary data. The data elements have (type, length, value) format with registered types recorded at IANA Pretty Good Privacy (PGP) registry [i.32].

The opinion about binary encodings in clause 7.1.1.2.2.1 "Formatting and Parsing Complexity for S/MIME" would also be relevant to OpenPGP formatting and parsing.

OpenPGP provides simple guidance for canonicalization of text, and all other data is treated as binary data.

7.1.1.4 XML Security

7.1.1.4.1 High Level Description of XML Security

XML security specifications are generated by the W3C's XML Security Working Group [i.33] in the form of "W3C recommendations". The latest recommendations, published in in 2013, are described below - (the descriptions borrow heavily from the respective documents).

- XML Encryption Syntax and Processing Version 1.1 [i.11] specifies how to encrypt data and represent the result in XML. The result of encrypting data is an XML Encryption `EncryptedData` element that contains (via one of its children's content) or identifies (via a URI reference) the cipher data. The data may be in a variety of formats, including octet streams and other unstructured data, or structured data formats such as XML documents, an XML element, or XML element content.

- XML Signature Syntax and Processing Version 1.1 [i.10] provides integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere. An XML Signature may be applied to the content of one or more resources. Enveloped or enveloping signatures are over data within the same XML document as the signature; detached signatures are over data external to the signature element.

- XML Signature Properties [i.34] defines a namespace and three properties to be used in XML Signatures: a Profile Property (a URI) identifying how in the signature is to be used (e.g. constraining the choice of algorithms); a Role Property (a URI) specifying an application specific role for the signature; and an Identifier Property enabling use cases where a unique identifier needs to be associated with the signature.

NOTE: There are a variety of additional technical reports on "XML Security 2.0", but these have status of "WG Notes", and are not endorsed as "W3C Recommendations".

XML Encryption and XML Signatures can be applied to provide desired combinations of confidentiality, integrity, message authentication, and/or signer authentication.

7.1.1.4.2 Considerations for XML Security

7.1.1.4.2.1 CoAP identification of the XML Security media type

The output of XML encryption and XML signatures are represented in XML. XML parsers processing the XML will be able to identify the XML encryption elements and XML signature elements. Consequently, an oneM2M Protocol

binding can support transporting XML encryption and XML signatures provided the protocol can identify the XML media type.

XML uses the media type "`application/xml`" in the IANA Media Types registry [i.23].

CoAP identities the XML media type using the CoAP Content-Format ID "41" (see [i.26]).

7.1.1.4.2.2 Formatting, Parsing and Canonicalization Complexity for XML Security

XML formatting and parsing is relatively easy and well supported. However, the need for XML canonicalization [i.2] in XML Encryption and XML Signatures introduces significant complexity.

The JOSE use cases document [i.3] expresses this opinion on formatting and parsing complexity for XML Security:

> `In practice, however, XML-based secure object formats introduce similar levels of complexity to ASN.1 (e.g., due to the need for XML canonicalization), so developers that lack the tools or motivation to handle ASN.1 aren't likely to use XML security either."`

This quote should be interpreted as an opinion, rather than technical fact, but it worthy of consideration.

7.1.1.4.2.3 Canonicalization and XML Security

Canonical XML is specified in recommendation [i.2]. XML encryption and XML Signature convert all XML to the Canonical XML prior to applying cryptographic processes. See clause 7.1.1.4.2.2 for discussion on the impact of canonicalization on formatting and parsing complexity for XML security.

7.1.1.5 JSON Security

7.1.1.5.1 High Level Description of JSON Security

The IETF JSON Object Signing and Encryption (JOSE) working group [21] has been chartered to develop a secure object format based on JSON. The JOSE working group has published the following documents:

- JOSE use cases and requirements. [i.3]. The use cases include security tokens, OAuth, OpenID Connect, XMPP, emergency alerting, constrained devices (including object security for CoAP).
- JSON Web Algorithms (JWA) [i.35] registers cryptographic algorithms and identifiers to be used with JOSE specifications.
- [i.36] JSON Web Key (JWK) defines JSON-based data structures that represent a cryptographic key (JWK) and a set of JWKs (JWK set)."
- [i.37] JSON Web Encryption (JWE) represents encrypted content using JSON based data structures. The JWE cryptographic mechanisms encrypt and provide integrity protection for an arbitrary sequence of octets.
- [i.38] JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures. The JWS cryptographic mechanisms provide integrity protection for an arbitrary sequence of octets."

The JOSE cookbook [i.21] provides a representative set of examples of protecting content using JOSE.

The secured objects produced using JOSE specifications can use either a JSON serialization or a compact, URL-safe text serialization (intended for space constrained environments such as HTTP Authorization headers and URI query parameters).

7.1.1.5.2 Considerations for JSON Security

7.1.1.5.2.1 CoAP identification of the JSON Security media type

The output of JWK, JWE and JWS can use a JSON serialization. JSON parsers processing the JSON will be able to identify the JWK, JWE and JWS elements. Consequently, an oneM2M Protocol binding can identify JWK, JWE and JWS provided the protocol can identify the JSON media type.

JSON uses the media type "`application/json`" in the IANA Media Types registry [i.23].

CoAP identities the JSON media type using the CoAP Content-Format ID "50" (see [i.26]).

7.1.1.5.2.2 Formatting, Parsing and Canonicalization Complexity for JSON Security

The formatting and parsing complexity of XML is comparable to the formatting and parsing complexity of JSON; formatting and parsing is relatively easy and well supported. JWK, JWE and JWS do not use canonicalization, which makes the formatting and parsing of JWK, JWE and JWS less complex than formatting and parsing for XML Security (which requires canonicalization).

7.2 Group Authentication

7.2.1 Group Authentication Solution 1

Figure 7.2.2-1 illustrates the sequence of events when using the group authentication solution 1 for the static group which corresponding to Clause 6.1.1. In this description, "Entity Ax" or "Entity Ay" corresponds to either a CSE or AE in a group, "Entity B" corresponds to MN-CSE, and "Entity C" corresponds to IN-CSE.

Credential Configuration: The Provisioned Credential for M2M Security Association Establishment ($Kpsa_{xm}$, $Kpsa_{ym}$, $Kpsa_{mi}$, $Kpsa_{xi}$, and $Kpsa_{yi}$) and the corresponding Provisioned Credential for M2M Security Association Establishment Identifier, denoted $Kpsa_{xm}Id$, $Kpsa_{ym}Id$, $Kpsa_{mi}Id$, $Kpsa_{xi}Id$, and $Kpsa_{yi}Id$ are provisioned to both entities either with pre-provisioning or remote provisioning.

NOTE 1: The provisioning (by definition) uses mechanisms not specified by oneM2M. The remote provisioning is performed thanks to Security Bootstrap Frameworks described in clause 7.3 in TS0003 [i.8].

NOTE 2: Entity Ax, Entity Ay, Entity B, and Entity C shall be configured with the information needed for the authentication and identification during the Inner/Outer Group Authentication and Group Security Association Handshake. For instance, Entity Ax is configured with Entity B identity (IdB) associated with the Provisioned Credential for M2M Security Association Establishment identifier $Kpsa_{xm}Id$ to establish security context by using $Kpsa_{xm}$. Entity Ax is to use this identity IdB for Entity B authenticating using the related arguments. This identity is also used to route the (D)TLS exchange.

Inner Group Authentication:

- Each Entity Ax (or Ay) in a group and the Entity B perform a (D)TLS-PSK handshake [i.39] to establish a secure session:

Take Entity Ax as example:

  - The "psk_identity" parameter [i.39] is set to the value of the Provisioned Credential for M2M Security Association Establishment identifier $Kpsa_{xm}Id$ between the Entity Ax and the Entity B.
  - The "psk" parameter [i.39] is set to the value of the Provisioned Credential for M2M Security Association Establishment $Kpsa_{xm}$ between the Entity Ax and the Entity B.

- The Entity Ax and the Entity B authenticate each other by verifying Message Integrity Codes (MIC) which was generated using the symmetric key ($Kpsa_{xm}$) to make mutual authentication.

NOTE 3: Each Entity in a group shall perform inner group authentication with the Entity B.

NOTE 4: This is just an example to use (D)TLS-PSK here, it is not limited to (D)TLS-PSK only.

Outer Group Authentication: Due to the successful Inner Group Authentication, Entity B could be on behalf of all entities in the group to make mutual authentication with the Entity C.

- The Entity B and Entity C perform a (D)TLS-PSK handshake [i.39] to establish a secure session:
  - The "psk_identity" parameter [i.39] is set to the value of the Provisioned Credential for M2M Security Association Establishment identifier $Kpsa_{mi}Id$ between the Entity B and the Entity C.
  - The "psk" parameter [i.39] is set to the value of the Provisioned Credential for M2M Security Association Establishment $Kpsa_{mi}$ between the Entity B and the Entity C.
- The Entity B and Entity C authenticate each other by verifying Message Integrity Codes (MIC) which was generated using the symmetric key ($Kpsa_{mi}$) to make mutual authentication. After that, each Entity in the group has carried out the authentication with the help of the Entity B.
- The Entity C generates the M2M Group Secure Connection Key for each Entity in the group. Especially, in the IN's database, there is a mapping between the identifier of MN (e.g. IdB) and each Entity's identifier (e.g. IdAx) in the group. For instance, Entity C generates Ks_x for Entity Ax from the Provisioned Credential ($Kpsa_{xi}$) between the Ax and C, a random number (Rand) and the identifier of Ax (IdAx). Moreover, Entity C sets the Group Secure Connection Key Identifier (Ks_xId) equal to $Kpsa_{xi}Id$, and stores the M2M Group Secure Connection Key (Ks_x) and the related Identifier (Ks_xId).

  NOTE 5: The derivation of the M2M Group Secure Connection Key, e.g. Ks_x := HMAC-SHA-256($Kpsa_{xi}$, "oneM2M Group Secure Connection Key derivation" || Enrolee-Ax-ID || Rand), where $Kpsa_{xi}$ is the value of the Provisioned Credential between the Enrolee Ax and the IN; Rand is a random number generated by the Entity C; Enrolee Ax's CSE-ID or AE-ID (Enrolee-Ai-ID), which shall be encoded to an octet string according to UTF-8 encoding rules as specified in IETF RFC 3629 [i.40] and apply Normalization Form KC (NFKC) as specified in [i.41]; HMAC-SHA-256 is defined in RFC 2014 [i.42].

Group Security Association Handshake: The Group Security Association Handshake enables the establishment of the M2M Group Secure Connection Key (e.g. Ks_x) and the associated Key Identifier (e.g. Ks_xId) shared between each Entity Ax (or Ay) in the group and the Entity C.

Take Entity Ax as example:

- Entity C sends the random number (Rand) to Entity B.
- Entity B forwards the Rand to the Entity Ax.
- Entity Ax generates the M2M Group Secure Connection Key (Ks_x) from the Provisioned Credential ($Kpsa_{xi}$) between the Entity Ax and the Entity C, the random number (Rand) and the identifier of Ai (IdA), sets the Group Secure Connection Key Identifier (Ks_xId) is $Kpsa_{xi}Id$, and stores the M2M Group Secure Connection Key (Ks_x) and the related Identifier (Ks_xId). After that, Entity Ax sends the Service Request Message to Entity B.
- Entity B forwards the Service Request Message to Entity C.

  NOTE 6: The Service Request Message is sent by the entity in a given group when the entity requires the IN to provide the services.

- Upon the Service Request Message, Entity B forwards this message to IN.

  NOTE 7: After each Entity in a group (e.g. Ax) sets up security association with IN using the same key (e.g. Ks_x), each Entity could be seen as authenticated when the security association works, since the mutual authentication between each Entity (e.g. Ax) and the IN is fulfilled through the proof of possessing the M2M Group Secure Connection Key (e.g. Ks_x).

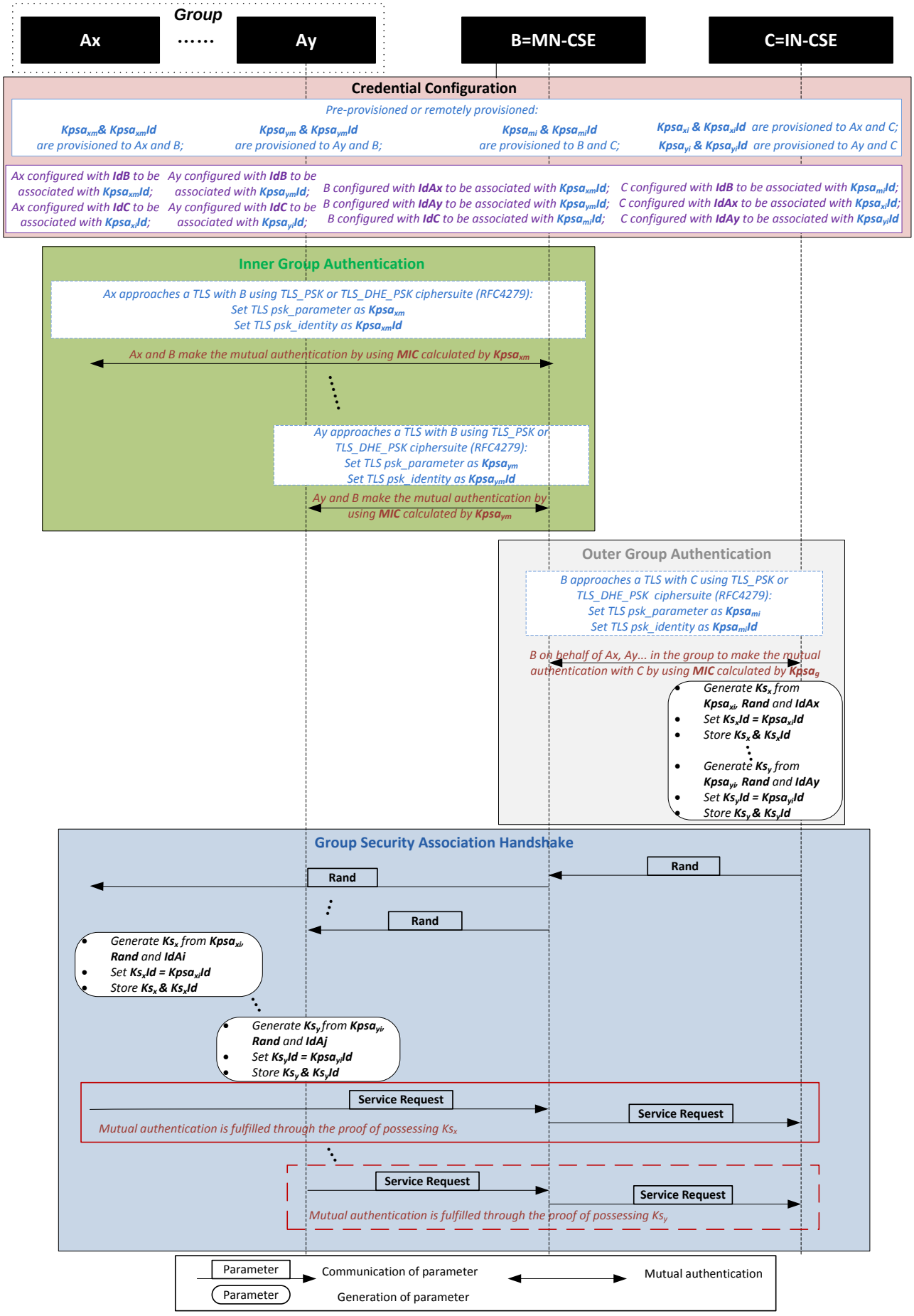

Group
Ax
.....
Ay
B=MN-CSE
C=IN-CSE
Credential Configuration
Pre-provisioned or remotely provisioned:
Kpsa_xm& Kpsa_xmId are provisioned to Ax and B;
Kpsa_ym & Kpsa_ymId are provisioned to Ay and B;
Kpsa_mi & Kpsa_miId are provisioned to B and C;
Kpsa_xi & Kpsa_xiId are provisioned to Ax and C;
Kpsa_yi & Kpsa_yiId are provisioned to Ay and C
Ax configured with IdB to be associated with Kpsa_xmId;
Ax configured with IdC to be associated with Kpsa_xiId;
Ay configured with IdB to be associated with Kpsa_ymId;
Ay configured with IdC to be associated with Kpsa_yiId;
B configured with IdAx to be associated with Kpsa_xmId;
B configured with IdAy to be associated with Kpsa_ymId;
B configured with IdC to be associated with Kpsa_miId;
C configured with IdB to be associated with Kpsa_miId;
C configured with IdAx to be associated with Kpsa_xiId;
C configured with IdAy to be associated with Kpsa_yiId
Inner Group Authentication
Ax approaches a TLS with B using TLS_PSK or TLS_DHE_PSK ciphersuite (RFC4279):
Set TLS psk_parameter as Kpsa_xm
Set TLS psk_identity as Kpsa_xmId
Ax and B make the mutual authentication by using MIC calculated by Kpsa_xm
Ay approaches a TLS with B using TLS_PSK or TLS_DHE_PSK ciphersuite (RFC4279):
Set TLS psk_parameter as Kpsa_ym
Set TLS psk_identity as Kpsa_ymId
Ay and B make the mutual authentication by using MIC calculated by Kpsa_ym
Outer Group Authentication
B approaches a TLS with C using TLS_PSK or TLS_DHE_PSK ciphersuite (RFC4279):
Set TLS psk_parameter as Kpsa_mi
Set TLS psk_identity as Kpsa_miId
B on behalf of Ax, Ay... in the group to make the mutual authentication with C by using MIC calculated by Kpsa_g
Generate Ks_x from Kpsa_xi, Rand and IdAx
Set Ks_xId = Kpsa_xiId
Store Ks_x & Ks_xId
Generate Ks_y from Kpsa_yi, Rand and IdAy
Set Ks_yId = Kpsa_yiId
Store Ks_y & Ks_yId
Group Security Association Handshake
Rand
Rand
Rand
Generate Ks_x from Kpsa_xi, Rand and IdAi
Set Ks_xId = Kpsa_xiId
Store Ks_x & Ks_xId
Generate Ks_y from Kpsa_yi, Rand and IdAj
Set Ks_yId = Kpsa_yiId
Store Ks_y & Ks_yId
Service Request
Service Request
Mutual authentication is fulfilled through the proof of possessing Ks_x
Service Request
Service Request
Mutual authentication is fulfilled through the proof of possessing Ks_y
Parameter
Communication of parameter
Mutual authentication
Parameter
Generation of parameter

Figure: 7.2.1-1: The sequence of events When using Group Authentication Solution 1

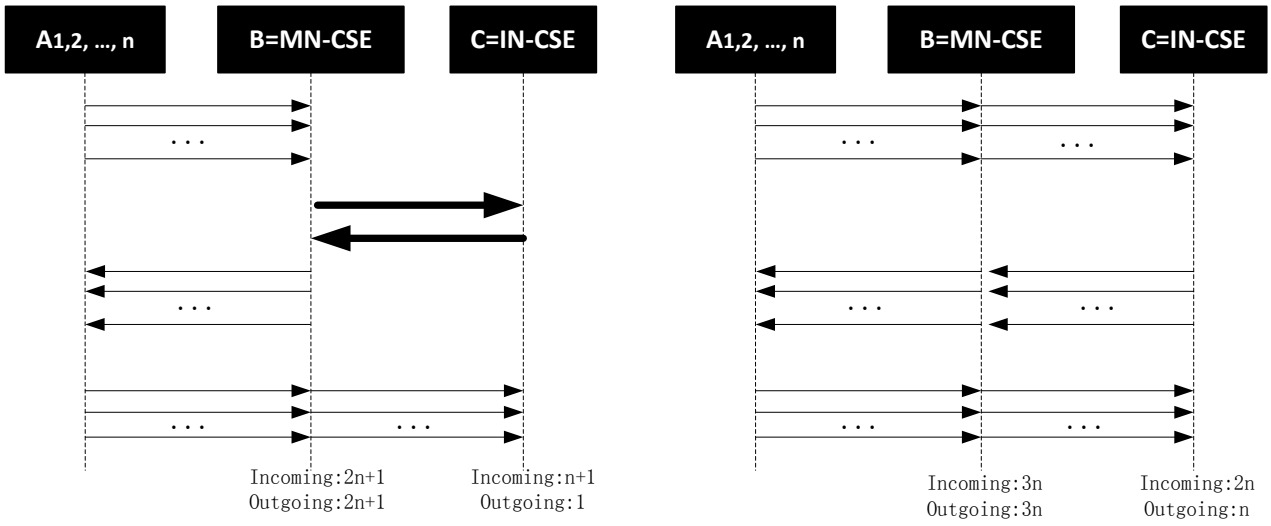
The benefits of using Group Authentication Solution 1 are twofold:

1) End to end security and privacy between individual M2M Device and M2M Server is protected even with a third party M2M Gateway standing in between.

Each device shares with the server a secret key Ksx after authentication, which is used to protect the end to end communication. These secret keys are unknown to the gateway and other devices in the group.

2) Number of messages the server and the gateway have to handle is decreased compared to the case where the gateway acts as a transparent proxy.

As illustrated in Figure 7.2.1-2, if group authentication solution 1 is used, to fulfil mutual authentication for all the n devices in the group, the server needs to handle n+1 incoming messages and only one outgoing message, the gateway needs to handle 2n+1 incoming messages and 2n+1 outgoing messages. If the gateway acts as a transparent proxy, the server needs to handle 2n incoming messages and n outgoing messages, the gateway needs to handle 3n incoming messages and 3n outgoing messages. As long as n>1, group authentication solution 1 can always leads to a message number deduction for both the server and the gateway.



(a) Group Authentication solution 1

(b) Gateway acting as transparent proxy

Figure: 7.2.1-2: Comparison of message-handling when using Group Authentication Solution 1

7.3 A Solution for providing security of data “at-rest”

7.3.1 General procedure for hosting and accessing secure data

The generic procedure for hosting and accessing secure data is illustrated in Figure 7.3.2-1 and described as follows:

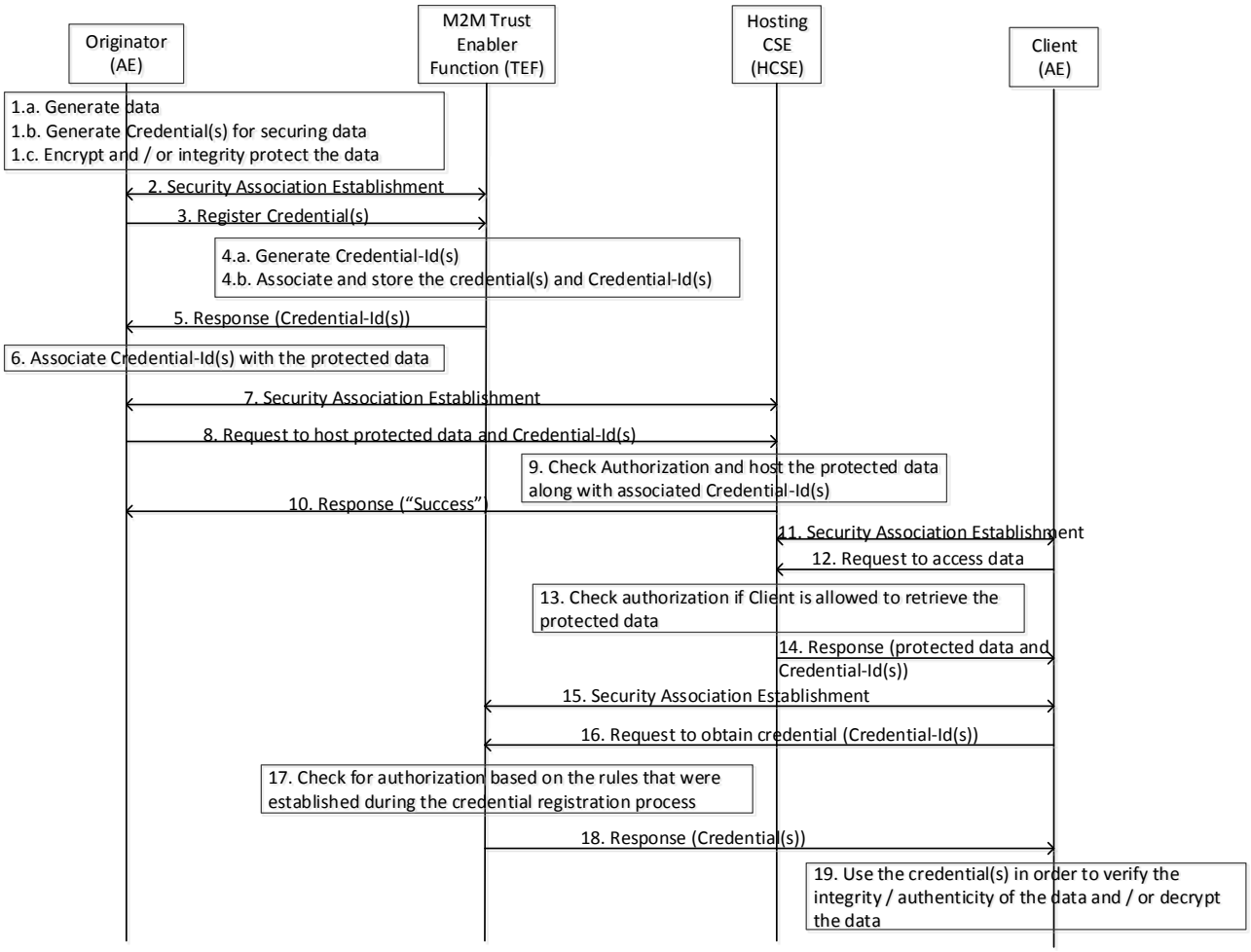


Figure 7.3.2-1: Hosting and accessing secure data

1. The Originator of data (AE) would like to provide protection (integrity /authenticity and / or confidentiality) to the data that it generates and hosts onto a Hosting CSE.

   a) Data is generated by an originator

   b) It also generates appropriate credential(s) for protecting the data. Alternatively, the Originator may not generate the credential(s) but instead request credential(s) from a Credential Registry.

   c) The originator then either encrypts, integrity protects or performs both to the data using the credential(s) that it generated

   d) The data originator may optionally provide access control rule(s) associated with the credential(s)

2. The Originator establishes a secure communications channel with an M2M Trust Enabler Function (TEF)

3. The Originator then requests to register the credential(s) that it generated with the TEF along with optional access control rule(s), that are used to govern an entity's access to the credentials

4. The TEF registers the credential(s)

   a) The TEF generates or creates unique Credential-Id(s)

   b) The TEF associates the Credential-Id(s) with respective credential(s) and stores them securely

   c) The TEF sets an expiration to the credential registration

   d) The TEF also associates access control rules to the credential(s) that were either provided by the originator or created by the TEF.

5. The TEF sends a response containing the associated Credential-Id(s) to the Originator
6. The Originator then associates the Credential-Id(s) with the respective protected data
7. The Originator establishes a secure communications channel with a Hosting CSE (HCSE)
8. The Originator sends a request to register and host the protected data along with the associated Credential-Id(s) with the HCSE.
9. The HCSE checks to ensure that the Originator is authorized to perform registration of protected data. If the Originator is deemed to be authorized then the HCSE registers and hosts the data along with the associated Credential-Id(s)
10. The HCSE sends a response indicating "Success"
11. A Client application (AE) that would like to access the secure data produced by an Orignator AE, performs a security association establishment process with the HCSE
12. The Client AE would like to retrieve the protected data that was created by the Originator AE and therefore sends a request to the HCSE
13. The HCSE checks to see if the Client AE is authorized to perform retrieval of the protected data
14. If the Client AE has been authorized then the HCSE sends the protected data along with the Credential-Id(s) associated with the protected data to the Client AE.
15. The Client AE establishes a secure communications channel with the TEF
16. The Client AE requests to retrieve the credential(s) by sending the respective Credential-Id(s) to the TEF
17. The TEF checks to see if the Client AE has been authorized to retrieve the credential(s) identified by the respective Credential-Id(s)
18. If the Client AE has been authorized then the TEF responds with the corresponding credential(a) using the secure channel
19. The Client AE uses the credential(s) in order to verify the integrity / authenticity and / or decrypt the data if it has been encrypted using appropriate credential(s)

It should be noted that the TEF function may be implemented as part of the CSE or as part of a centralized entity (e.g. MEFy)

7.3.2 Bootstrapped procedure for providing data security

7.3.2.1 Overall Description

This section provides detailed description on mechanisms that can be employed for providing security of data both "at-rest" and "in-transit" in an end-to-end manner. If an entity (e.g. AE) has a trust relationship with an M2M Trust Enabling Function (TEF), then that trust can be leveraged in order to generate credentials for integrity / authenticity and or confidentiality of data. Here, the term "data" is used in a generic manner may represent an oneM2M resource, which may be a system resource (e.g. <*mgmtObj*>) resource or application generated data or instances of application generated data (e.g. <*contentInstance*>).

7.3.2.2 Detailed Description

Figure 7.3.2.2-1 illustrates a message between entities involved in data generation, data hosting, and generating credentials for data security as well consumer of protected data. The functional roles associated with each entity is further described below:

- **Originator (AE1):**
    - Has a trust relationship with a M2M Trust Enabler Function (TEF) and associated credentials
    - Ability to generate data security-specific credentials based on a bootstrapping procedure with the TEF
    - Ability to request registration of Credential-Id with a TEF and associated ACPs
    - Generates protected (integrity and or encrypted) data or data instances (contentInstances) using the data-security-specific credentials
    - Has a trust relationship with a Hosting (HCSE) and establish a secure communications channel with the HCSE
    - Ability to request hosting of protected data onto a HCSE
- **Hosting CSE (HCSE):**
    - Has a trust relationship with an Originator AE1 and also a separate trust relationship with Client application AE2 and associated credentials
    - Ability to process request by Originator AE1 for hosting of protected data and the ability to host protect data and data instances.
    - Ability to process a Retrieve request by a Client application AE2 for protected data or data instances
- **Trust Enabler Function (TEF):**
    - Has a trust relationship with an Originator AE1 and also a separate trust relationship with Client application AE2 and associated credentials
    - Ability to process request by Originator AE1 for performing a bootstrapping process and generate data-security-specific credentials.
    - Ability to perform Credential-Id registration request from Originator AE1
    - Ability to process a Retrieve request by a Client application AE2 for data-security-specific credential(s)
- **Client (AE2):**
    - Has a trust relationship with a M2M Trust Enabler Function (TEF) and associated credentials
    - Has a trust relationship with a HCSE and associated credentials in order to establish a secure connection with it.
    - Ability to request a “Retrieve” operation on protected data hosted on a HCSE
    - Ability to request “Retrieve” operation for credentials identified by a Credential-Id with a TEF
    - Ability to verify authenticity / integrity of data or data instances and or the ability to decrypt the data or data instances obtained from HCSE using the credentials that were retrieved from TEF.

The detailed messaging steps are provided below:

0. The bootstrapping process described within TS-0003 (Release 1) specifications is further enhanced by using the bootstrapping process in order to derive “data security”-specific credentials. The shared credential KpmId / Kpm between an AE and a MEF or a TEF is used to generate session credentials KeId / Ke, as described in TS-0003 (Release 1). The credential Ke is then used to generate data security-specific credentials between the AE and the TEF. Similar mechanisms may be used if the bootstrapping process is used between an AE and CSE leveraging existing security association between AE and CSE. The security association identified by KpsaId / Kpsa as described in oneM2M TS-0003. The Kpsa is then used instead of Ke. Similarly, KmId / Km that is used for establishing security association between an AE and MAF may be used to generate data security credentials

between AE and MAF. It must be noted that a more preferable approach is for the AE and TEF to generate the data security credentials.

1. A Salt that is a random value that is used as part of the key generation mechanism may have been shared during the bootstrapping process or may computed as a hash value of the initial communications between the AE and the TEF during the bootstrapping process. The Salt may be a cryptographic representation of the channel that is bound between AE1 and the TEF. The channel may be a secure connection established using TLS or DTLS. As part of the "Data Security Credential Generation" process, the AE1 (Enrolee) and the TEF (Enrollment Target) generates data security credentials using the Ke, (Ke_AE1-TEF refers the Ke that is associated between AE1 and TEF) as the master key in order to generate the data security master key, K_AE1_TEF_data_sec_master. Alternatively, if the target is a MAF, then the Km would be used as the master key for the generating the data security master key. An Example of data security Key Generation using RFC 5869 [i.43], where the Enrollee is AE and the Enrollment Target is TEF, is provided below:

   K_AE1_TEF_data_sec_master = HMAC-Hash (Salt, Ke_AE1_TEF)

   T(0) = empty string (zero length)

   Once the K_AE1_TEF_data_sec_master has been generated it is then used for Key Expansion in order to generate unique data authenticity and data confidentiality keys. In some cases, only a single key is generated if the data authenticity as well confidentiality is provided by an algorithm such as the AEAD (e.g. AES-CCM or AES-GCM)

   K_AE1_TEF_data_auth = T(1) = HMAC-Hash (K_AE1_TEF_data_sec_master, T(0) | "Data Authenticity and Integrity"| 0x01)

   The K_AE1_TEF_data_auth key is used for providing data authenticity and data integrity and is referred to as data authenticity or data integrity key.

   K_AE1_TEF_data_conf = T(2) = HMAC-Hash (K_AE1_TEF_data_sec_master, T(1)| "Data Confidentiality Key"|0x02)

   The K_AE1_TEF_data_conf key is used for providing data confidentiality and is referred to as the data confidentiality key.

   In certain cases, the Kpsa_AE1_CSE1 (which is the Kpsa between AE1 and CSE1) is used instead of Ke_AE1_TEF, and the process described above is used to generate unique keys for data security protection, namely, data authentication / integrity and data confidentiality. Kpsa is used if a CSE is used by the AE as the data security credential registry in place of a TEF.

   In certain other cases, only a single session key, K_AE1_TEF_data_auth_conf that is generated from Ke or Kpsa or Kpm is used for providing both data authenticity as well as data confidentiality, when used with AEAD class of algorithms.

2. The TEF performs a similar process of key generation. The negotiation of the algorithm, key generation mechanisms, types and number of keys to be generated etc.. are assumed to have been performed during step 0, when the bootstrapping process was carried out between AE1 and TEF.

3. AE1 generates content / data, where each instance of the content / data may be protected by unique set of data authenticity and data confidentiality keys or all instances of the content is protected by a single data authenticity key and by a single data confidentiality key. An example key generation where only a single data authenticity and a data confidentiality key for a container that may contain multiple content instances is generated is shown below:

   K_AE1_Container-x_data_auth = HMAC-Hash (K_AE1_TEF_data_auth, "Data Authenticity and Integrity"| "Container-x"| Nonce or creationTime) and

   K_AE1_Container-x_data_conf = HMAC-Hash (K_AE1_TEF_data_conf, "Data Confidentiality"| "Container-x"| Nonce or creationTime)

   Alternatively for each instances of content within a container, a unique set of keys may be generated:

   K_AE1_ContentInstance-x_data_auth = HMAC-Hash (K_AE1_TEF_data_auth, "Data Authenticity and Integrity"| "Container-x"| Nonce or creationTime) and

K_AE1_ContentInstance-x_data_conf = HMAC-Hash (K_AE1_TEF_data_conf, "Data Confidentiality"| "ContentInstance"| Nonce or creationTime)

The content is encrypted and or integrity protected using the above generated keys. For encrypting the content a random IV may be generated by AE1 and uses it along with the encryption algorithm and the content (data) in order to generate the encrypted content (R1-EC, the encrypted resource).

If content instance are being encrypted separately, then each content instance has a unique confidentiality key and a new IV is generated each time an encryption process is carried out generating an encrypted content instance; therefore each content instance has an associated separate encrypted content instance.

For integrity protecting or adding authenticity to content / data, a random Nonce along with Time component is used in order to generate an Authentication Tag (AT) of the content.

If each content instance is being protected separately then each content instance has an associated AT. In the case of providing data authenticity in certain cases, it may be preferable to use a single key for generating each individual ATs.

The encrypted content may be represented as the modified oneM2M container or <*contentInstance*> resource as depicted in Figure 7.3.3.2-2. Alternatively, The encrypted content R1-EC that is created may be based upon the JSON Web Encryption (JWE) specified in **Error! Reference source not found.**, while the R1-AT that is created may be based upon the JSON Web Signature specified in **Error! Reference source not found.**. The appropriate algorithms may be represented in the form as specified in the JSON Web Algorithms (JWA) standards **Error! Reference source not found.**.

As a general case, each key that is generated and used is associated with a unique Credential-Id. The Credential-Id may be generated by the AE1 or provided by the TEF to AE1. In certain cases, the Credential-Id may carry characteristics of the content id or content instance id and the type of credential. An example of a Credential-id may be of the form:

K_AE1_Container-x_data_conf-Id@TEF.com, which is associated with the key, K_AE1_Container-x_data_conf

4. AE1 registers the Credential-Id with the TEF. The AE1 Requests to create a Credential resource identified by a Credential-Id and associated Access Control Policies (ACP). The Credential-Id may alternatively be generated and provided by the TEF in order to avoid collision of Credential-Ids that are associated with the TEF. Collisions of Credential-Ids may be avoided if a hash is carried out of the generated id by AE1:

   H1 = Hash (K_AE1_Container-x_data_conf-Id)

   Credential-Id = H1@TEF.com

5. The TEF checks to ensure that AE1 has been authorized to register the credentials with the TEF also verifies the Credential-Id as well as optional Cryptographic Parameters (CryptoParams) that may have been included. The TEF then creates a <*credential*> resource type and populates it with the necessary attributes, such as, the <*accessControlPolicy*> values.

6. The TEF sends a Response that indicates successful creation of the Credential (<*credential*>) resource to AE1.

7. AE1 and HCSE establishes a secure connection as per TS-0003 security solutions using (D)TLS.

8. AE1 Requests to create a secure resource R1, that is encrypted (encrypted resource denoted as: R1-EC) and integrity protected using R1-AT. AE1 also provides the necessary CryptoParams, Credential-Id. The CryptoParams details the type of algorithm to be used, the length of the keys, mechanisms on how the protection is to be carried out etc. The Credential-Id may be part of the CryptoParams or may be sent as separate child resource associated with R1. AE1 also includes the associated ACPs. The ACP may be integrity protected.

9. HCSE verifies the request and checks to ensure that AE1 has been authorized to create a resource at HCSE

10. HCSE responds with a success message.

11. At some point a client (AE2) would like to retrieve R1. The AE2 and HCSE performs a mutual authentication and sets up a secure connection using (D)TLS.

12. AE2 sends a Request to "Retrieve" the resource R1, to HCSE. Mechanisms involved in discovering R1 is outside the scope and it is assumed that the AE2 is able to discover the location of a secure version of R1.

13. HCSE verifies the authorization of whether AE2 is allowed to perform a retrieve operation using the information within the ACP that was created by AE1.

14. The HCSE sends a Response containing the R1-EC, EC-AT as well as R1-CryptoParams. The R1-EC may be represented using e.g. JSON-based notation, JWE, while the EC-AT may be represented using JWS and the R1-CryptoParams may be represented using JWA. Alternatively, both the EC-AT and R1-EC may be represented as JWE especially if the algorithm used for encryption and integrity protection is based on AEAD algorithm (e.g. AES-GCM, AES-CCM). Alternatively, the encrypted content, R1-EC as well the R1-AT may be represented as an oneM2M resource along with the appropriate CryptoParams.

15. If the Credential-Id was included as part of the CryptoParams, AE2 extracts the Credential-Id from it.

16. The AE2 and TEF performs mutual authentication and sets up a secure connection using (D)TLS.

17. AE2 sends a Request message to the TEF in order to perform a retrieve operation by including the Credential-Id as the resource-id within the message. The Credential-Id may be sent using e.g. JSON-based notation such as JWK or using the oneM2M resource structure. The AE2 may also extract a Salt or Nonce from the CryptoParams that is associated with the resource R1 (data). AE2 may also send the Salt or Nonce along with the Credential-Id in order to retrieve resource-specific credentials.

18. TEF verifies authorization of AE2 based on the ACP that was created by the AE1 during the credential registration process in 2.

19. If AE2 is authorized to retrieve, then the TEF computes the resource-specific credentials sends the credentials over a secure channel to the AE2. In addition the TEF may also send usage info, on how the credentials may be used and the associated algorithms that are to be used. The AE2 may already have possession of the usage info that it obtained from the HCSE as part of the CryptoParams. However, in certain cases where a container may contain a number of contentInstance resources and each with its own encrypted contentInstance and authentication tags and associated credentials then the TEF may be able to provide additional guidance on how the credentials may be used in order to verify contentInstance integrity as well how it may be used to decrypt the contentInstances. If the Salt is sent by then the TEF may generate the:
    K_AE1_TEF_data_sec_master, which is then provisioned to the AE2. The AE2 uses the K_AE1_TEF_data_sec_master in order to generate container-specific or contentInstance-specific credentials. The mechanisms to generate K_AE1_TEF_data_auth and K_AE1_TEF_data_conf and associated container or contentInstance-specific credentials may follow similar to the mechanisms detailed earlier.

    Alternatively, the TEF may provision either K_AE1_TEF_data_auth and / or K_AE1_TEF_data_conf to the AE2, which then generates the container-specific or contentInstance-specific credentials. In other cases, the TEF may only provision just the container-specific or contentInstance-specific credential(s) to the AE2. The AE2 does not perform any key generation since it is provisioned with the keys and thereby limiting the AE2 from having cryptographic access to specific container or contentInstance(s). It should be noted that for each of the keys the associated Nonce(s) and or creation time associated with the container or content instances may have to be provided to the TEF. In certain cases, when the AE1 performs registration of the credential process in step 4, the AE1 may include the CryptoParams associated with the Credential-Id with the TEF.

    From a performance and security perspective, the more preferred approach may be for the TEF to only provision the K_AE1_TEF_data_auth and / or K_AE1_TEF_data_conf to the AE2. It should be noted that all the credentials have an associated life-time associated with it. After the expiration of lifetime, new credentials may have to be generated.

20. Using the credentials AE2 verifies the integrity using R1-AT and decrypts R1 using the provisioned or generated container or contentInstance credentials.

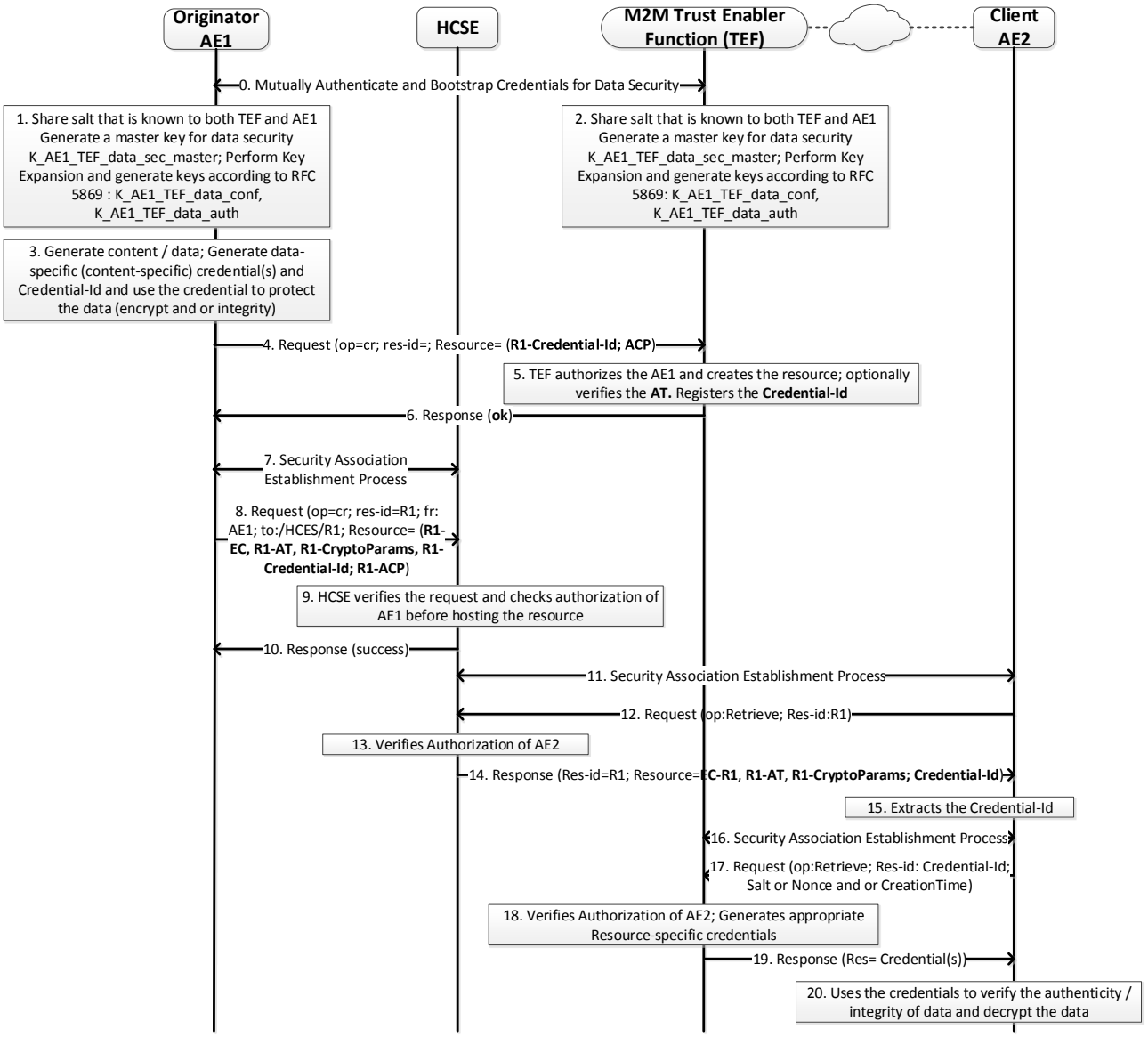


Figure 7.3.2.2-1: Message flow depicting hosting and retrieval of protected content

An example of a protected <*contentInstance*> is depicted in Figure 7.3.2.2-2. The contentInstance has an *encryptedContent* as a child resource and associated *cryptoParams*, the cryptographic parameters that are to be used in order to be able to decrypt the contentInstance. In addition, an *authenticationTag* or *digitalSignature* resource and associated *cryptoParams*, the parameters that are to be used in order to be able to verify the authentication tag or digital signature are added to the existing *contentInstance* resource.

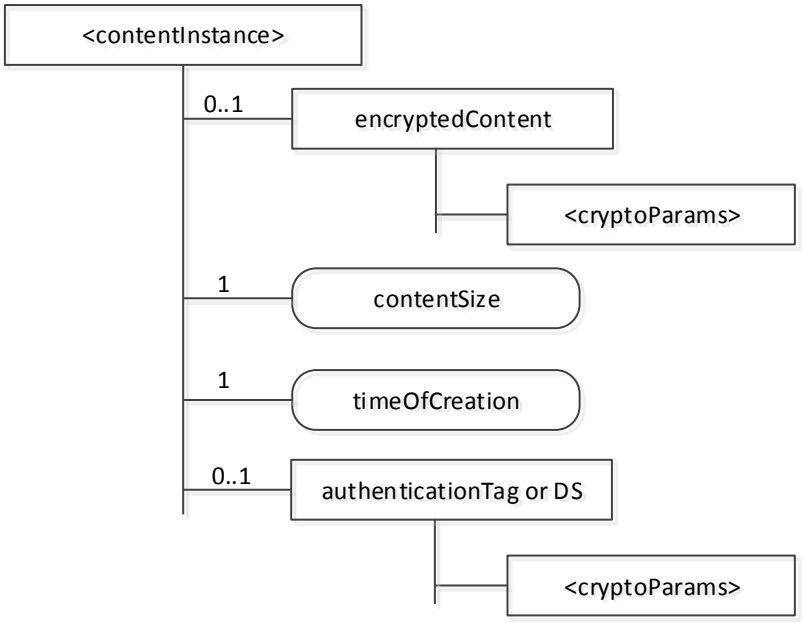


Figure 7.3.2.2-2: Resource structure of a protected <*contentInstance*>

7.4 A Solution for providing End-to-End Message Authentication using Symmetric Key

7.4.1 End-to-End Security Credential(s) Generation Process

7.4.1.1 Overall Description

This section provides description on mechanisms that can be employed for generation of credential(s) that are to be used for end-to-end security. Based on security requirements associated with an Entity (e.g. AE), appropriate end-to-end security credentials may be generated.

7.4.1.2 Detailed Description

Table 7.4.1.2-1 illustrates the end-to-end security requirements that have been determined and provided by the entity (e.g. AE1) or determined based on a security profile associated with AE1. The table illustrates that AE1 provides or requires the following:

- End-to-End message authentication using a Message Integrity Code (MIC)
- Also provides for or requires the ability to verify the integrity of data in-transit
- End-to-End message confidentiality in transit and thereby also provides for end-to-end data confidentiality in-transit.

Security Requirement	**End-to-End**
Message Originator Authenticity / Integrity	Message Integrity Code (MIC): 256 bits

Message Re-play protection	Nonce / Random / Time Component
Non-repudiation capability	None
Message Confidentiality	Encryption: 256 bit encryption
Confidentiality of Data in Transit	Encryption: 256 bit encryption
Confidentiality of Data at Rest	None
Integrity of Data in Transit	Message Integrity Code (MIC): 256 bits
Integrity of Data at Rest	None

Table 7.4.1.2-1 End-to-End Security Requirements

Based on the higher-level requirements appropriate end-to-end credentials may be generated using either bootstrapping process (preferable approach) or by using pre-provisioned end-to-end credentials. Illustrated in Figure 7.4.1.2-1 is a high-level key generation process.

As part of the "Generation of End-to-End Key Generation" Phase, the enrolee and the enrolment target generates End-to-End credentials using the Kpsa as the master key in order to generate the End-to-End master key. If the enrolee is an AE, and the Enrolment Target is a CSE, then an end-to-end master credential, Ke2e_AE_CSE_master is generated. An Example of End-to-End Key Generation using RFC 5869 is provided below:

Ke2e_AE_CSE_master = HMAC-Hash (Salt, Kpsa_AE_CSE)

Using the generated end-to-end master key, the associated end-to-end message authentication and or end-to-end message confidentiality keys are generated in the following manner:

T(0) = empty string (zero length)

Ke2e_AE_CSE_msg_auth = T(1) = HMAC-Hash (Ke2e_AE_CSE_master, T(0) | "E2E Message Authentication Key"| 0x01)

Ke2e_AE_CSE_msg_conf = T(2) = HMAC-Hash (Ke2e_AE_CSE_master, T(1)| "E2E Message Confidentiality Key"|0x02)

This process is repeated by each Enrolee and associated Enrolment Target based on a unique Enrollee-EnrolmentTarget_Ke2e_master that is shared between the Enrolee and the Enrolment Target (e.g. AE and CSE specific end-to-end keys).

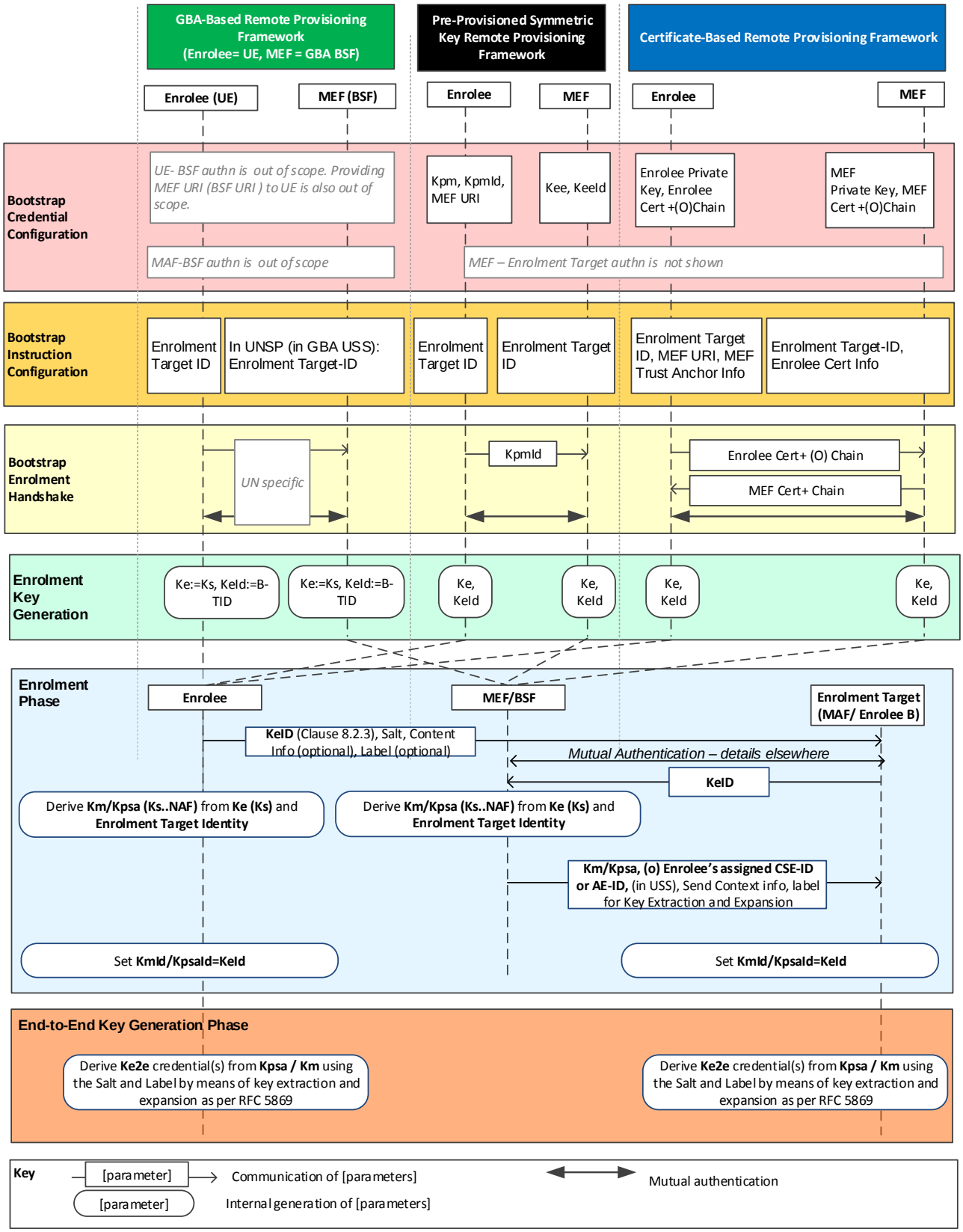


Figure: 7.4.1.2-1: End-to-End Key Generation Phase

An example list of generated keys illustrated in Table 7.4.1.2-2:

Security	Symmetric Keys Generated and Used	Parameters
Message Originator Authenticity / Integrity	Ke2e_EntityA_EntityN_msg_auth	None
Message Re-play protection	Ke2e_EntityA_EntityN_msg_auth	Nonce / Time / Seq#
Non-repudiation capability	N/A	N/A
Message Confidentiality	Ke2e_EntityA_EntityN_msg_conf	IV
Confidentiality of Data in Transit	Ke2e_EntityA_EntityN_msg_conf	IV
Confidentiality of Data at Rest	None	None
Integrity of Data in Transit	Ke2e_EntityA_EntityN_msg_auth	Nonce / Time / Seq#
Integrity of Data at Rest	None	None

Table 7.4.1.2-2: Depicting example keys that generated and associated parameters

7.5 Proposal for determining detailed Security Requirements, Features and associated Algorithms

7.5.1 Security Determination Process

7.5.1.1 Overall Description

This section provides description on mechanisms that can be employed for determining the appropriate security requirements, features, and algorithms and associated credentials in order that the security mechanisms that are employed are adequate but at the same time efficient for the type of service offered by the entity and also does not drain valuable resources from the entity.

7.5.1.2 Detailed Description

Table 7.5.1.2-1 illustrates Entity Profile (EP) associated with an entity (e.g. AE1). The EP may be obtained or provided explicitly during the registration phase. It may be obtained as part of the M2M Subscription Profile associated with an entity. The EP provides a very high-level security requirement of the Entity. It may be used to derive and generate a more granular security requirement as illustrated in Figure 7.5.1.3-4. However, since the profile of the entity does not provide details of the device, the selection of the features may not be appropriate from a performance perspective.

Entity Profile	Values
Class of Service	Healthcare
Type of Service	Real-time

Impact	Critical (Life and Limb)
Security Level	High

Table 7.5.1.2-1 Entity Profile of an Entity (e.g. AE1)

An example Device Profile (DP) is illustrated in Table 7.5.1.2-2. Using the DP as well as the EP, a more appropriate security requirements and features may be generated. So, a low-powered, low-memory device that only provides a service that requires "low" Security, then the security function(s), the algorithms selected and the key sizes may be selected appropriately. E.g. the message authentication mechanism selected may be HMAC-SHA1 with 160 bit keys whereas an entity with more processing and memory and requiring a higher security would be provisioned with 256 bit keys that may be used with HMAC-SHA2 mechanism.

Device Capability	**Values**
Processing Capability	900 MHz
RAM	500 Kb
Flash	1MB
Battery	5.0 Micro-W/MHz
Wireless Capability	Bluetooth, WiFi
Sleep Mode	Sleep / Deep-Sleep
Secure Environment	Yes
OS / version	Android / Kitkat

Table 7.5.1.2-2 Device Profile of an Entity (e.g. AE1)

An example Security Profile (SP) is illustrated in Table 7.5.1.2-3. Even if both the EP and DP are available, in certain cases, it may not be enough to determine, for example, whether end-to-end security is required or if data at-rest security is required or not. In such cases, it would be preferable to be provided with an SP.

Security Requirement	**Within Security domain (at HCSE or RCSE)**	**End-to-End**
Message Originator Authenticity / Integrity	High	Very High
Message Re-play protection	High	Very High
Non-repudiation capability	Low	Low
Message Confidentiality	Medium	Medium
Data Confidentiality in Transit	High	High
Data Confidentiality at Rest	Medium	High
Data Integrity in Transit	High	Very High
Data Integrity at Rest	Medium	Very High

Service Availability	High	Very High
Data Availability	High	Very High

Table 7.5.1.2-3 Security Profile of an Entity (e.g. AE1)

An example security features that have been determined based upon the information provided in the SP, DP and EP is illustrated in Table 7.5.1.2-4.

Entity ID	**Security Features**	**Within Security Domain**			**End-to-End**	
		Algorithms	**Sizes**	Protocol(s)	**Algorithms**	**Sizes**
AE1	Message Originator Authenticity / Integrity	HMAC-SHA-2	256 / 512	(D)TLS, JWS	HMAC-SHA-2	256 / 512
	Message Replay Protection	Nonce	256	N/A	Timestamp / Nonce + Sequence Number	256 bits
	Non-Repudiation	None		N/A	None	
	Message Confidentiality	AES	112	(D)TLS	AES	192
	Confidentiality of Data in Transit	AES	192	(D)TLS, JWE	AES	192
	Confidentiality of Data at Rest	AES	256	N/A	AES	256
	Integrity of Data in Transit	HMAC-SHA-2	256	(D)TLS, JWS	HMAC-SHA-2	256
	Integrity of Data at Rest	HMAC-SHA-512	512	N/A	HMAC-SHA-512	512
	Authentication Mechanism	Symmetric Key	256	(D)TLS	Symmetric Key	256
	Authentication Process	Direct				
	Presence of Secure Element	YES				

Table 7.5.1.2-4: An example of detailed security features determined based on SP, EP and DP

8 Release 2 End-to-End Security and Rationale

8.1 Overview of Release 2 End-to-End Security Features

The analysis in the preceding clauses were used to guide the specification of Release 2 features for End-to-End Security and Group Authentication, which include the following:

- **End-to-End Security of Data (ESData):** provides an interoperable framework for protecting data that ends up transported using oneM2M reference points, in order that transited CSEs do not need to be trusted with that data.
- **End-to-End Security of Primitives (ESPrim):** provides an interoperable framework for securing oneM2M primitives so CSEs do not need to be trusted with the confidentiality and integrity of the primitive.
- **End-to-End Security Certificate-based Key Establishment (ESCertKE):** provides an interoperable framework for two end-points to use certificates for establishing a secret symmetric key called pairwiseESCertKE from which symmetric keys are derived for use in other end-to-end security frameworks such as End-to-End Security of Data (ESData) or End-to-End Security of Primitives (ESPrim).
- **Generic MAF Security Framework:** describes common details and procedures used in the MAF-based SAEF, MAF-Based ESPrim and MAF-Based ESData protection options.

8.2 Release 2 End-to-End Security of Data (ESData)

8.2.1 End-to-End Security of Data (ESData) Overview

Overview: End-to-End Security of Data (ESData) provides an interoperable framework for protecting data that ends up transported using oneM2M reference points, in order that so transited CSEs do not need to be trusted with that data. The data to be protected is called the *ESData Payload*. ESData payload could typically compose all or part of an attribute value (e.g. *content* attribute value of a <*contentInstance*> resource) or a primitive parameter (e.g. a signed, self-contained access token communicated in a request primitive to obtain dynamic authorization).

ESData assumes that there is a single *ESData Source End-Point* applying the ESData processing to the Payload to obtain an *ESData Envelope* containing the secured data and necessary headers, with one or more *ESData Target End-Points* applying the ESData processing to the Envelope to extract the verified data. The Payload is composed of plaintext (which is to be encrypted and integrity protected) and associated authenticated data (which is to be integrity protected only).

There is no inherent restriction on which entities can be Source End-points and Target End-Points; these end-points may be entities inside a oneM2M system (that is, AEs and CSEs) or entities outside of a oneM2M system (for example, entities which are part of a system that interworks with oneM2M).

Corresponding End-to-End Security and Group Authentication requirements: If described in the terminology of the End-to-End Security Framework (ESF) described in clause 6.2, ESData satisfies the requirements (clause 6.2.2.2) of:

- ESF security session type ESF-S1 “One-way Single envelope security session” at the ESF Security Layer,
- ESF Target Data class 3 “*content* attribute”.

Group authentication requirements (see clause 6.1.2) are applicable to protection of messages. Consequently, group authentication requirements are not addressed by ESData.

8.2.2 End-to-End Security of Data (ESData) Functional Architecture

Functional architecture details for ESData are found in TS-0003 [i.8].

The following ESData security classes are provided:

- **Encryption only**: (see Note 1) which offers confidentiality and integrity protection. This payload is protected using symmetric keys, and these symmetric keys are established using one or more of the following:
    - Symmetric keys otherwise established with the target end-points. In this case, the source end-point can be authenticated – unless the symmetric key was shared with multiple target end-points.
    - Target end-points certificate. When target end-point certificate are used, the target end-point cannot authenticate the source end-point.

      NOTE 1: Strictly speaking, this class provides encryption and integrity protection, but this name aligns usage in protocols such as JSON Web Encryption (JWE) and XML-Encryption which can provide both encryption and integrity protection.
- **Signature only**: which offers source authentication, integrity protection and (when asymmetric digital signatures are used) non-repudiation. This uses either symmetric keys based MIC or asymmetric digital signatures verified using source end-point certificates.
- **Nested Sign-then-Encrypt**: This is used in cases where encryption is required in addition to source authentication and/or non-repudiation using a source end-point certificate. A digital signature(s) on the payload is signed first, and then encryption is applied to combination of the payload and digital signature.

ESData supports using multiple credentials protecting a single payload unit.

The ESData protection options for each ESData Security Class are shown in Table 8.2.2-1.

Table 8.2.2-1: ESData protection options

ESData Security Class	ESData Protection Option	Key Management	Source Verification	Non-Repudiation
Encryption only	Encryption using Provisioned Symmetric ESData Key	Provisioned Symmetric Key	Symmetric	-
	Encryption using MAF	MAF	Symmetric	-
	Encryption using Target End-Point Certificate	Certificate	-	-
Signature only	MIC using Provisioned Symmetric ESData Key	Provisioned Symmetric Key	Symmetric	-
	MIC using MAF	MAF	Symmetric	-
	Digital Signature using End-Point Source End-Point Certificate	Certificate	Certificate	Certificate
Nested-Sign-then-Encrypt	Digital Signature using End-Point Source End-Point Certificate followed by any combination of Encryption-Only Protection Options	Provisioned Symmetric Key(s) and/or MAF(s) and/or Certificate(s) for Encryption. Certificate for Signature	Certificate	Certificate

ESData protection options provide the range of options available for Security Associated Establishment Framework (SAEF) defined in Release 1: a Provisioned Symmetric Key option; an option using an M2M Authentication Function for operation-phase key distribution; and an option using certificates.

Associated Changes to Release 1 Security functionality:

- The Remote Security Provisioning Frameworks (RSPFs) in Release 1 are updated in Release 2 to support provisioning symmetric keys for use in ESData and ESPrim. See clause 8.6.1 “Changes to Remote Security Provisioning Frameworks (RSPFs)” for further details.
    - The RSPFs in Release 1 are (independently) updated in Release 2 to support provisioning certificates – these certificates can be used in ESData protection options using certificates.
- A generic MAF Security Frameworks is specified to support distributing symmetric keys for using in ESData. See clause 8.5. “Release 2 MAF Security Framework” for further details.

8.3 Release 2 End-to-End Security of Primitives (ESPrim)

8.3.1 End-to-End Security of Primitives (ESPrim) Overview

Overview: End-to-End Security of Primitives (ESPrim) provides an interoperable framework for securing oneM2M primitives so CSEs do not need to be trusted with the confidentiality and integrity of the primitive. ESPrim provides mutual authentication, confidentiality, integrity protection and a freshness guarantee (bounding the age of ESPrims).

The primitive to be secured is called the *inner primitive*, and the primitive which is used to transport a secured inner primitive is called the *outer primitive.* The inner primitive is protected using encryption and integrity protection which using a symmetric key *sessionE2EPrimitiveKey* as input. The *sessionESPrimKey* is derived from a *pairwiseESPrimKey*, established between the Originator and Receiver, and a *receiverE2ERandObject* and *originatorE2ERandObject*. The receiverE2ERandObject can be generated on a per-Originator basis, or can be shared and distributed according to the principles of Group Authentication in clause 6.1. The originatorE2ERandObject is generated on a per-Receiver basis.

Corresponding End-to-End Security and Group Authentication requirements: If described in the terminology of the End-to-End Security Framework (ESF) described in clause 6.2, ESPrim satisfies the requirements (clause 6.2.2.2) of:

- ESF security session type ESF-Sm "Two-way Multi-envelope security session" at the ESD Security Layer,
- ESF Target Data class 1 "Entire request or response".

Group authentication requirements (see 6.1.2) are addressed by ESPrim.

8.3.2 End-to-End Security of Primitives (ESPrim) Functional Architecture

The credential management aspects and data protection aspects for ESPrim are specified in the present clause. Clause 11.3.2, TS-0003 [i.8] specifies the transport of ESPrim.

The inner primitive is protected using encryption and integrity protection which takes a symmetric key *sessionE2EPrimitiveKey* as input. The *sessionESPrimKey* is derived from a *pairwiseESPrimKey*, established between the Originator and Receiver, and a *receiverE2ERandObject* and *originatorE2ERandObject*.

Sequence of events for ESPrim consists of three main phases:

A. **Establishing pairwiseESPrimKey**, via
   - Pre-provisioning,
   - A Remote Security Provisioning Frameworks (RSPF) – The RSPF text is updated to support ESPrim, see clause 8.6.1).
   - End-to-End Security Certificate-based Key Establishment (ESCertKE), see clause 8.4.
   - MAF Security Framework – which was specified to support MAF-Based ESPrim, see clause 8.5.

B. **Establishing sessionESPrimKey at the Originator.** The Receiver shall select to either (a) pre-generate a receiverE2ERandObject which is distributed for used by multiple Originators for establishing sessionESPrimKey, or (b) generate a unique receiverE2ERandObject upon request form the Originator. In the former case, the Receiver regularly changes the receiverE2ERandObject element within an *e2eSecurityCapabilities* attribute of the Receiver's <*remoteCSE*> resource on a CSE to which it is registered. The originator attempts to retrieve this pre-generated receiverE2ERandObject first, and if no receiverE2ERandObject element is present then the Originator requests a unique receiverE2ERandObject directly from the Receiver. The Originator then generates its originatorE2ERandObject, and sessionESPrimKey is derived from a pairwiseESPrimKey, receiverE2ERandObject and originatorE2ERandObject

C. **Securing a primitive exchange**. Object-security technology is used to form an ESPrim Object containing the encrypted and integrity protected inner primitives, with appropriate header parameters. The ESPrim Object is transported in a Notify request or response according to whether the inner primitive is a request or response. The ***Security Info*** parameter of the Notify request or response indicates that the message contains an ESPrim Object.

8.4 Release 2 End-to-End Security Certificate-based Key Establishment (ESCertKE)

8.4.1 End-to-End Security Certificate-based Key Establishment (ESCertKE) Overview

Overview: End-to-End Security Certificate-based Key Establishment (ESCertKE) provides an interoperable framework for two end-points to use certificates for establishing a secret symmetric key called *pairwiseESCertKE* from which symmetric keys are derived for use in other end-to-end security frameworks such as End-to-End Security of Data (ESData) or End-to-End Security of Primitives (ESPrim).

Corresponding End-to-End Security and Group Authentication requirements: Within the context of the End-to-End Security Framework describes in clause 6.2, ESCertKE satisfies the requirements (clause 6.2.2.2) of the ESF security session type ESF-Sm "Two-way Multi-envelope security session" at the ESF Security Layer.

ESCertKE is used for establishing credentials, while group authentication assumes that credentials are already established. Consequently, group authentication requirements are not addressed by ESCertKE.

Further Rationale: ESCertKE is proposed to enable efficient secure end-to-end communication between end points that can mutually authentication using certificates. While primitives could be secured individually using certificates, this adds significant overhead to each primitive, and for this reason, ESPrim does not support using certificates to directly secure the primitive. There are advantages to, establish a symmetric key using the certificates, and then secure the data objects or primitives using the symmetric key. A M2M Enrolment Function (MEF) or M2M Authentication Function (MAF) can be used to facilitate establishing this key, but there are scenarios where relevant stakeholders prefer the MEF or MAF do not know the symmetric key. ESCertKE enables the ESPrim end-points to establishing a symmetric key directly – without involvement of an MEF or MAF.

Using certificates to secure individual data objects also adds significant overhead to each data object. However, certificate-based security of data objects needs to be support for cases with multiple Target ESData end-points. None-the-less, there are ESData scenarios where ESCertKE is used to establish symmetric keys for securing the data object. Examples of the latter include cases where multiple data objects are being communicated between two ESData end-points.

8.4.2 End-to-End Security Certificate-based Key Establishment (ESCertKE) Functional Architecture

The ESCertKE messages and associated processing for ESCertKE are specified in TS-0003 [i.8]. The transport details for the ESCertKE Procedure are specified in TS-0001 [i.6].

The entities in the ESCertKE reference model are the *ESCertKE Initiating End-Point* which initiates the procedure and the single *ESCertKE Terminating End-Point* with which the ESCertKE Initiating End-Point intends to establish a pairwiseESCertKE.

The *ESCertKE Procedure* consists of the ESCertKE Initiating End-Point and ESCertKE Terminating End-Point exchanging a sequence of *ESCertKE Messages* and apply processing based on those ESCertKE Messages, as shown in in Figure 8.4.2-1 "ESCertKE Procedure message flow. The ESCertKE Messages contain TLS v1.2 [i.12] messages. If the ESCertKE Procedure is successful, then the ESCertKE Initiating End-Point and ESCertKE Terminating End-Point export a pairwiseESCertKE based on the parameters exchanged in the ESCertKE Messages.

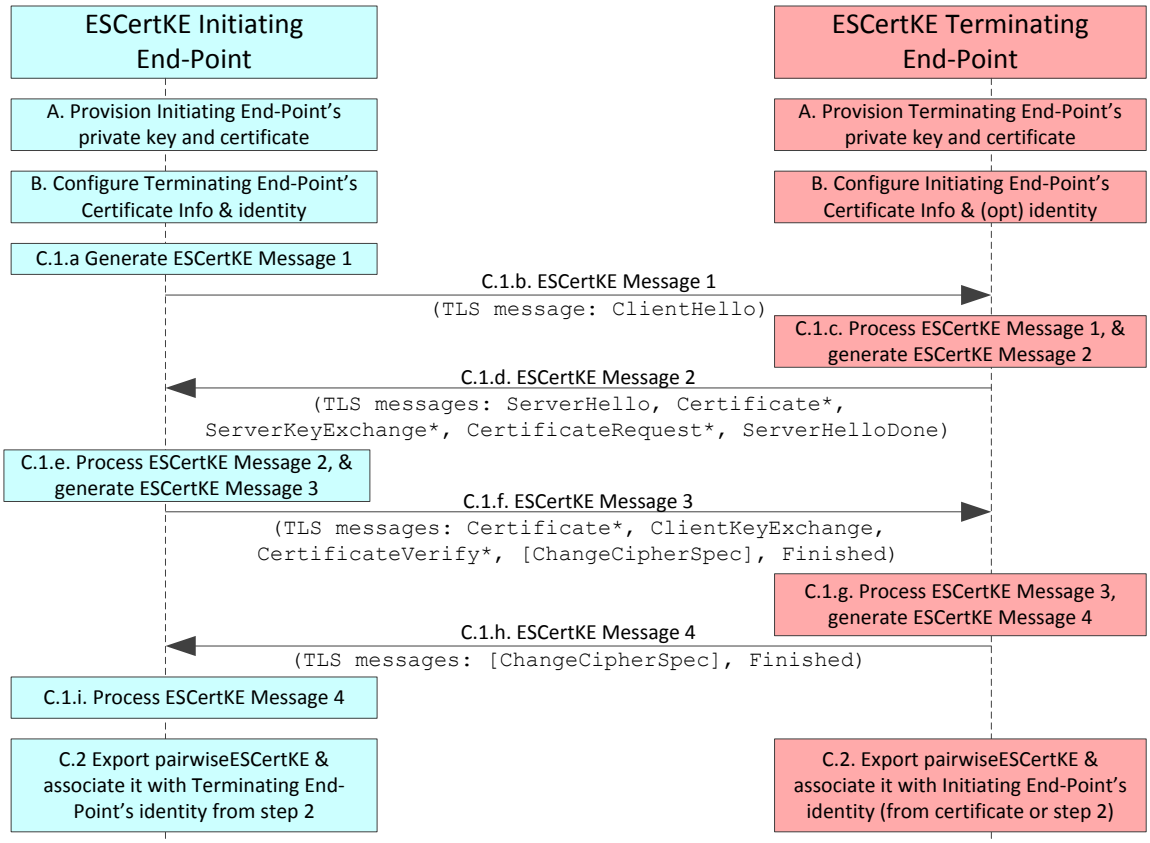


Figure 8.4.2-1: ESCertKE Procedure message flow

The ESCertKE messages are transported in a Notify request or response. The ***Security Info*** parameter of the Notify request or response indicates that the message contains an ESCertKE message.

There is no inherent restriction on which entities can be an ESCertKE Initiating End-point; these end-points can be entities inside a oneM2M system (that is, AEs and CSEs) or entities outside of a oneM2M system (for example, entities which are part of a system that interworks with oneM2M).

The only restriction on entities which can be ESCertKE Terminating End-Points is that the ESCertKE Terminating End-Point is able to receive the unsolicited ESCertKE Message initiating the ESCertKE Procedure.

8.5 Release 2 MAF Security Framework

8.5.1 MAF Security Framework Overview

Release 2 includes a generic MAF Security Framework describing common details and procedures used in the MAF-based Security Frameworks; in the present specification these frameworks include:

- The MAF-Based Security Association Establishment Framework (SAEF).
- The MAF-Based End-to-End Security of Primitives (ESPrim) Framework.
- The MAF-based End-to-End Security of Data (ESData) Framework.

These frameworks use a MAF to provide authentication and distribution of symmetric key for use by a Source End-Point initiating establishing the symmetric key, and one or more Target End-Points. The technical details are based on the text describing the Release 1 MAF-Based SAEF. Table 8.5.1-1 "Mapping of Generic MAF Framework Roles to specific MAF-Based Framework Roles" describes the mapping of Source End-Point and Target End-Point to roles in the specific MAF-Based Frameworks, and the allowed number of Target End-Points.

Table 8.5.1-1: Mapping of Generic MAF Security Framework Roles to specific MAF Security Framework Roles

MAF-Based Security Framework	Source End-Point	Target End-Point	Number of Target End-Points
Security Association Establishment Framework (SAEF)	Entity A	Entity B	1
End-to-End Security of Primitives (ESPrim)	Originator	Receiver	1
End-to-End Security of Data (ESData)	Source ESData End-Point	Target ESData End-Point	Any

Usage-constrained key derivation is included to ensure that symmetric keys distributed by a MAF can only be applied for its intended usage (e.g. for SAEF, ESPrim or an ESData protection option).

The MAF Security Framework also adds certificate-based authentication with the MAF – which is new to Release 2.

8.5.2 MAF Security Framework Functional Architecture

The generic MAF security framework specifies the following:

- **MAF Credential Configuration** describing the credentials and associated information provisioned to an End-Point and the MAF to enable the End-Point to use the services of the MAF.
- **MAF KmId Retrieval Procedure**: If a CSE or AE is remotely provisioned with a symmetric key for use with the MAF, then this procedure is used to trigger the MAF to retrieve Km from the MEF. The MAF then provides the End-Point with the KmId to be used for subsequently authentication with the MAF.
- **MAF Handshake Procedure**: establishing a mutually-authenticated TLS or DTLS session between an End-Point and a MAF.
- **MAF Key Registration Procedure**: The Source End-Point and MAF perform the MAF Handshake procedure. The Source End-Point may provide a list of possible Target End-Point(s) to the MAF. The MAF and Source End-Point establish a symmetric key and symmetric key identifier, and negotiate information about the credential.
- **MAF Key Retrieval Procedure**: The Target End-Point and MAF perform the MAF Handshake procedure. The Target End-Point provides the symmetric key identifier and SUID to the MAF. The MAF provides the symmetric key and associated information to the Target End-Point.

The general sequence for using these procedures is shown in Figure 8.5.2-1and described as follows:

1. Each of the Source End-Point and Target End-Point(s) separately establish credentials for mutual authentication with the MAF, and are configured with the URIs for the MAF Key Registration Procedure and MAF Key Retrieval Procedure. If a symmetric key is remotely provisioned for mutual authentication with the MAF, then the Source End-Point is also provided with the MAF KmId Retrieval URI.
2. If the Source End-Point is remotely provisioned for mutual authentication with the MAF, then the Source End-Point performs the MAF KmId procedure to trigger the MAF to retrieve the Source End-Point's Master Credential from the MEF. The MAF then provides the Source End-Point with the KmId to be used for subsequently authentication with the MAF.

The same procedures is performed by a Target End-Point which is remotely provisioned for mutual authentication with the MAF.

3. The Source End-Point performs the MAF Key Registration procedure to establish a symmetric key and corresponding identifier. The Source End-Point also provides the Security Usage Identifier (SUID) limiting the scope of the credential by identifying the security feature (SAEF, ESPrim, ESData). This procedure includes the MAF Handshake procedure for mutual authentication of the Source End-Point and MAF.

4. The Source End-Point provides, to the Target End-Point(s), the symmetric key identifier established in the MAF Key Registration procedure. The details of this step depend on the security feature as identified by the SUID.

5. The Target End-Point performs the MAF Key Retrieval procedure, to retrieve the symmetric key and corresponding information. This procedure includes the MAF Handshake procedure for mutual authentication of the Target -End-Point and MAF.

6. The symmetric key is used in the security protocol between the Source End-Point and Target End-Point.

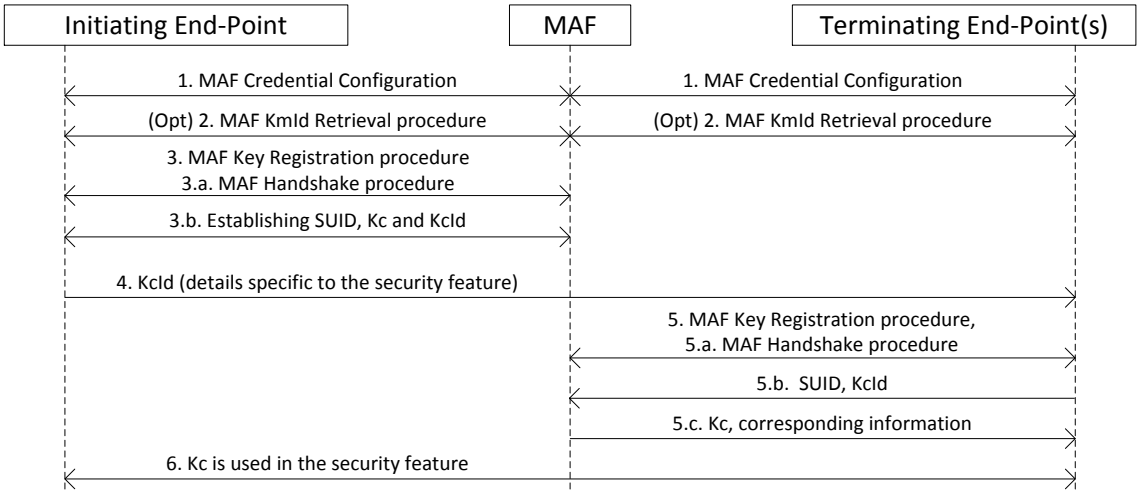


Figure 8.5.2-1: The sequence of events when using the MAF Security Framework as part of a security feature

8.6 Changes to Release 1 Features in Release 2

8.6.1 Changes to Remote Security Provisioning Frameworks (RSPFs)

The Remote Security Provisioning Frameworks (RSPFs) in Release 1 are updated in Release 2 to support provisioning symmetric keys for use in ESData protection options using a Provisioned Symmetric Key and ESPrim using remotely provisioned symmetric key. Usage-constrained key derivation is included to ensure that symmetric keys provisioned by an RSPF can only be applied for its intended usage (e.g. for SAEF, ESPrim or an ESData protection option).

The RSPFs in Release 1 are (independently) updated in Release 2 to support provisioning certificates which can be used in ESData and ESCertKE.

8.6.2 Changes to Security Association Establishment Frameworks (SAEFs)

The changes to the SAEFs do not introduce new functionality, but rather represent changes to accommodate defining a generic MAF-based procedures which can be used with SAEF, ESData and ESPrim. Some SAEF-specific details are still included in the MAF-Based SAEF description.

9 Conclusions and recommendations

The present document offers an overview of the use cases, requirements, architecture proposals and available solutions for End-to-End Security and Group Authentication.

Some of the contents have been normalized as Release 2 Technical Specification, as described in clause 8. Others may be used to facilitate future normative work resulting in oneM2M Technical Specifications.

Annex A: Problem Statement for needing End-to-End Data Security

A.1 Introduction

The general protection afforded to data / content is only while the content is “in transit” between two “trusted” entities by means of transport-layer protocol (e.g. TLS / DTLS). There is no notion of protection of content while hosted at an

entity (at rest). The implication is that the content (object) protection, if at all performed, must be done at the application layer. The problem with the application layer content protection approach is multi-fold:

1. Applications do not provide a uniform mechanism for application data / content protection. Application layer protection only provides for the actual application data protection and not the resources associated with the oneM2M service layer.
2. Service Layer resources are not protected.
3. A separate application layer protocol for securing content must be performed, which may be far more cumbersome
4. In certain scenarios, in order for a service layer to be able to provide value-added services, the content may have to be stored in its un-encrypted form

Use-case 1

Illustrated in **Error! Reference source not found.** is a use-case where there are 4 entities involved: an AE1, a Hosting CSE (CES1), an IN-CSE and a hacker application or even a non-malicious function. The AE1 created a resource within a CSE1 at service layer which stores attributes and content / content instances. In this use case, two attributes are provided as examples: Attribute 1 and Attribute 2. It is assumed that the AE1 and CSE1 have mutually authenticated one another and used a secure communications channel prior to performing the "create resource" operation. At some point, a hacking application exploits a vulnerability within the CSE1 through an IN-CSE. It is possible that the hacking application may be able to reach the CSE1 without having to go through an IN-CSE, however, because of the diversity of protocol support (e.g. open ports and services that run) on the IN-CSE, it may make a very good entry point for a hacking application. Once the hacker is able to access the CSE1, it steals the content stored within the AE1 resource. It is a classic attack without the need for much sophistication. One way to mitigate such an attack is to encrypt the whole disk or use encryption on a per-file basis. However, the content may have to be processed at the SL and decrypted at a transit node on the communication path, where the content becomes vulnerable to an attack. Another mechanism is to protect the content using JSON-based object signing and encryption mechanisms. But there is no framework to enable the use of such mechanisms for protecting SL resources. An additional issue is that, the CSE1's platform is not trustworthy and therefore secure processes may not be carried out. Also, if a root key is broken, then it exposes the data from all the AEs stored on the CSE1. In short, the security of an application data or user's confidential data is off-loaded to an entity that the user or application does not have much control of and the trustworthiness of the platform is based on the trust the user has of the SP. Additionally, when the CSE1 is de-commissioned the data remains within the CSE1 and protected only by a file-based encryption, which may be broken easily by an obsolete operating system protecting the resources. The threats can be summarised as follows:

- Unauthorized access by an entity to data / content at rest
- Unauthorized access to data / content by an intermediate node (e.g. CSE) during transit
- Unauthorized access to system specific resources during rest or transit

A summary of the issues is provided here:

- Lack of a confidentiality protection mechanism at the hosting entity (e.g. CSE), when the content is at rest.
- Lack of a mechanism to hide the content even from the CSE (e.g. a less-trustworthy CSE)
- Each hop (e.g. transit CSE) has an un-encrypted access to the content once the data comes through a TLS/DTLS tunnel, when transferring content to a Client.

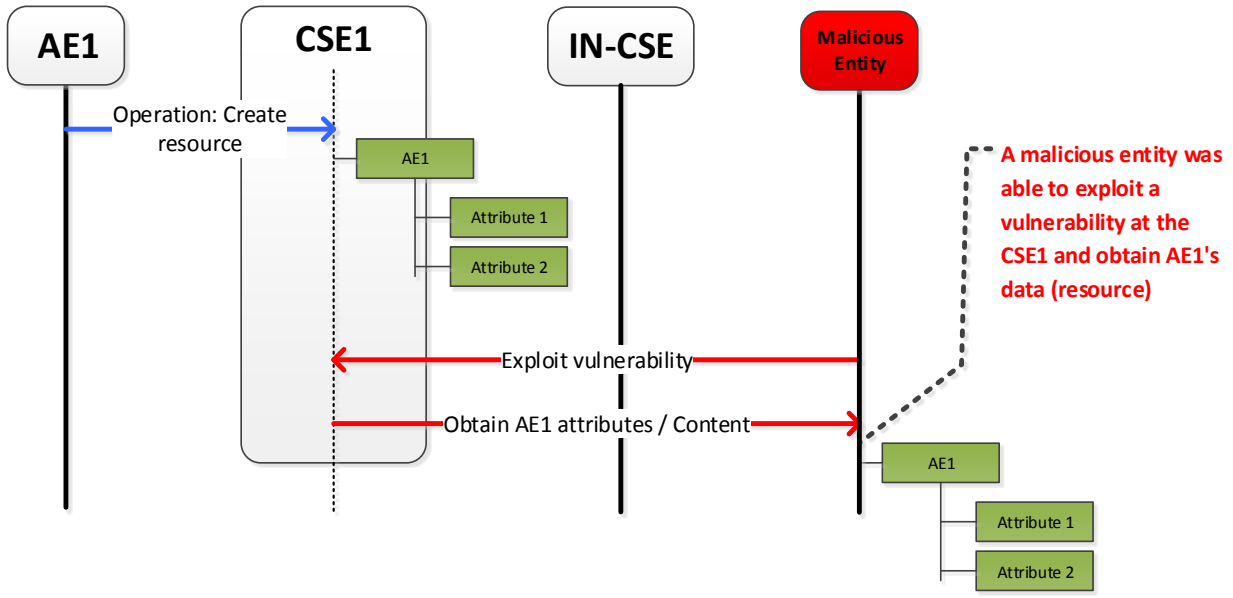


Figure 1: Un-authorized access to a data / content

Use-case 2

The use case illustrated in **Error! Reference source not found.**involves six entities: AE1 that generates the content, AE2 and AE3 are Client applications that consume the content produced by AE1. The content is hosted on CSE1 without any integrity protection. Similar to the above mentioned case, an attacker may exploit vulnerabilities either at the IN-CSE or CSE1 and modifies the resource and / or the resource structure, e.g. attributes and / or content(s). The figure illustrates a scenario where an attacker is able to perform an un-authorized modification of AE1's attribute, Attribute 1. AE2 that is subscribed to AE1's resource obtains the modified copy of the resource. In cases, where the resource obtained from AE1 is used to make critical decisions or operations by AE2 then it may have major ramifications. At a later point, the attacker deletes the Attribute 2 and adds new Attribute 3 and 4, essentially the attacker is not only changing the resource, but also the structure of the resource. An AE3 that is subscribed to the resource has a completely different resource tree than what was created by AE1. The threats can be summarised as follows:

- Unauthorized modification of data / content at rest
- Unauthorized modification of data / content by an intermediate node (e.g. CSE) during transit
- Unauthorized modification of system specific resources during rest or transit

Summary of the issues are listed below:

- There is no mechanism to provide integrity protection to a resource
- There is no mechanism to provide integrity protection to the structure of the resource
- There is no mechanism to provide integrity protection to system critical resources (e.g. ACPs, m2mSubscriptionProfile, managementObject resources etc.)

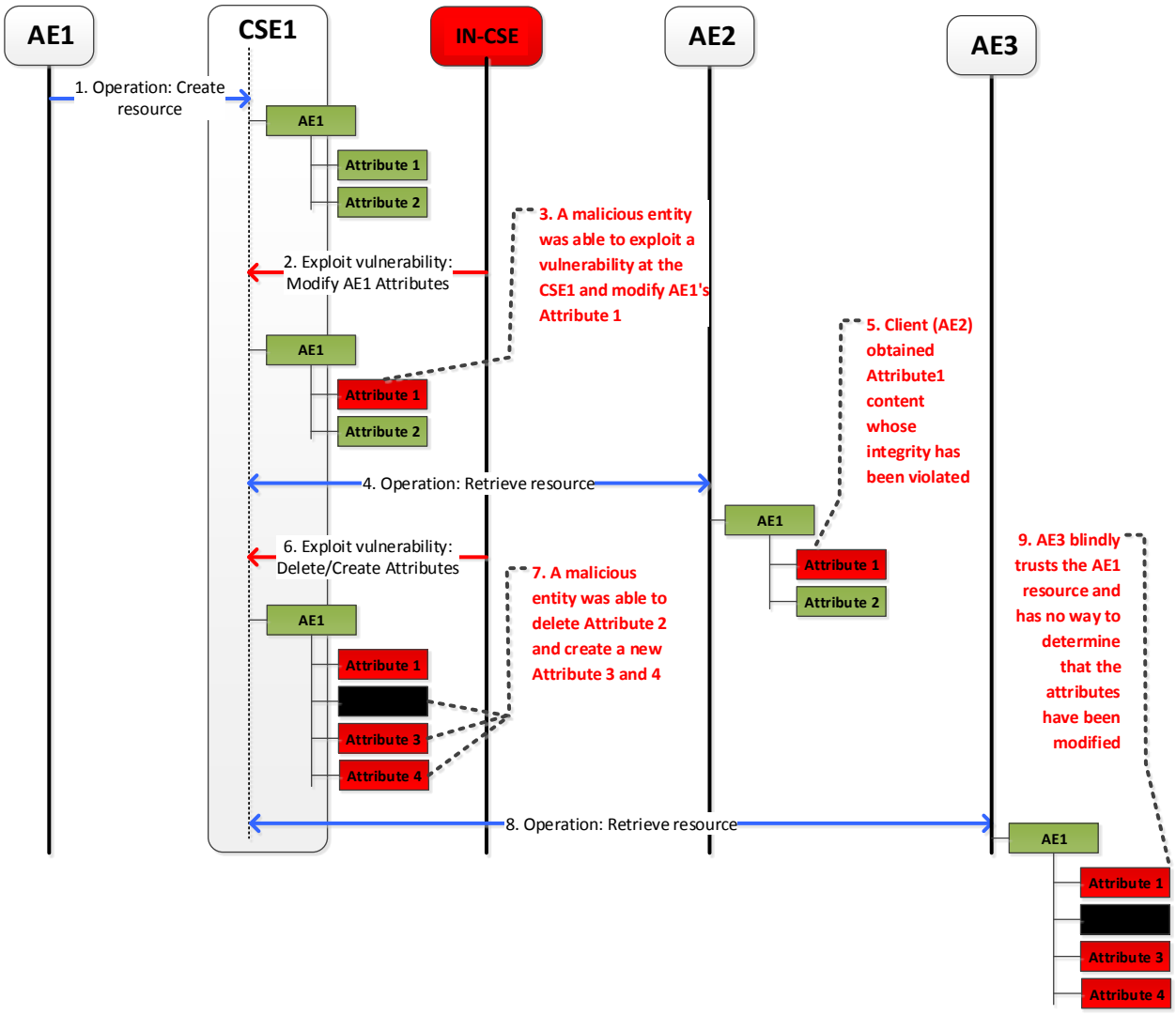


Figure 2: Un-authorized modification of data / content

Annex B: Use case for remote attestation

B.1 Description

In a oneM2M System, a simple application deployment where sensors provide a flow of messages from peripheric ADNs, relayed by MNs and finally up to a central IN is not always possible due to 1) lack of connectivity or 2) inefficient use of network bandwidth.

The lack of connectivity can be seen as a complete loss of connectivity or as the result of an unacceptably high network latency between the ADN and the IN. An example would be a car automation system. The sensor and the M2M Gateway are both part of the car. The processing has to be done inside the car such that the result is immediately available. When connectivity allows, the results can be shared with other vehicles or road infrastructure. For critical information (ie: speed), the receivers need a way to verify the correctness of the result (ie: a platoon of vehicles adjusting their speeds to match). This relates to OSR-022, which states a certain independence of Field Nodes when disconnected from the Infrastructure:

OSR-022	When some of the components of a M2M Solution are not available (e.g. WAN connection lost), the oneM2M System shall be able to support the normal operation of components of the M2M Solution that are available.	Implemented

		in Rel-1

Data aggregation is an example where the centralized approach leads to inefficient use of bandwidth. Although the IN is much more powerful than a sensor, it is not practical to aggregate millions (or billions) of data points from each individual ADN. A middleground would be to distribute the computation across the MNs and reduce the data flows to manageable sizes as they approach the collection point (IN). More explicitly, as messages flow from sensors through MNs, the MNs perform a computation based on these inputs (ie: max, average) and expose a smaller, virtual resource to the next MNs and ultimately to the IN. This distributed map-reduce helps makes the data collection practical. For critical applications (compute maximum of temperatures to trigger alarm), the receiver needs a way to verify the correctness of the result. This enables a secure fulfilment of the Semantics Mashup Requirements MSH-01 to MSH-05:

MSH-001	The oneM2M System shall provide the capability to host processing functions for mash-up.	Targeted for Release 2
MSH-002	The oneM2M System shall enable M2M Applications to provide processing functions for mash-up.	Targeted for Release 2
MSH-003	The oneM2M System itself may provide pre-provisioned or dynamically created processing functions for mash-up.	Targeted for Release 2
MSH-004	The oneM2M System shall be able to create and execute mash-ups based on processing functions.	Targeted for Release 2
MSH-005	The oneM2M System shall be able to expose mash-ups as resources e.g. virtual devices.	Targeted for Release 2

More generally, the previous examples show the value of an application design where the M2M Gateway can process the data from the originating sensor (constrained M2M device) on its way to the receiver (IN, another M2M Gateway, or M2M Device). For such a design to be useful, the receiver needs to be able to verify that the MN correctly processed the right data, thus providing end-to-end assurance of data and computation integrity.

B.2 Actors

The actors in this use case are the sensor (ADN-AE), the middle-node (MN-AE, or prover) and another node (ASN-AE, or verifier).

ADN-AE(s): One or multiple sensors act as the data source. They usually are constrained M2M Devices (ADN-AE). They are equipped with a TPM capable of digitally signing the data.

MN-AE: Is an application running on the MN that receives the inputs from the sensors and computes a function over them.

ASN-AE: Is another node that receives the output from the MN-AE and wishes an assurance that the result is correct (correct inputs and correct computation).

B.3 Pre-conditions

The sensors are equipped with TPMs (ie: they are tamper-proof and can cryptographically sign messages).

B.4 Normal flow

Setup

1. The Application Service Provider creates the application, deploys and configures it on MN-AE and on ASN-AE.

Operation

1. (Optional) Setup between ASN-AE and ADN-AE that might be needed to protect against replay attacks.
2. The sensor generates data, signs it, and sends it to the MN-AE.

3. The MN-AE computes the result of the function over the sensor data and a proof. It sends the result along with the proof to the ASN-AE.

4. The ASN-AE verifies the proof and based on the outcome accepts or rejects the result.

B.5 Potential requirements

The potential requirements match the security requirements SER-044 and SER-050:

SER-044	For M2M Application Service data, that are processed by an M2M Application B in a M2M entity (e.g. M2M Gateway) on its path from an originator A to the recipient M2M Application C, the oneM2M System shall provide means that enable the recipient to verify both: • integrity of the data received by the M2M Application B from the originator A and, at the same time • that the M2M Application B that has processed the data has not been compromized.	Targeted for Release 2
SER-050	The oneM2M System shall enable pre-defined conditions to be protected from unauthorized modification.	Targeted for Release 2

In the case of SER-050, the predefined conditions are the instructions that comprise the AE program itself. In that sense, satisfying SER-044 also satisfies SER-050.

History

Publication history		
V.2.0.0	30-Aug-2016	Publication