IEC 62351-9

Edition 1.0 2017-05

INTERNATIONAL STANDARD

colour inside

Power systems management and associated information exchange – Data and communications security –
Part 9: Cyber security key management for power system equipment

IEC 62351-9:2017-05(en)

IEC 62351-9

Edition 1.0 2017-05

INTERNATIONAL STANDARD

colour inside

Power systems management and associated information exchange – Data and communications security – Part 9: Cyber security key management for power system equipment

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 33.200

ISBN 978-2-8322-4220-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

POWER SYSTEMS MANAGEMENT AND ASSOCIATED INFORMATION EXCHANGE – DATA AND COMMUNICATIONS SECURITY –

Part 9: Cyber security key management for power system equipment

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62351-9 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
57/1838/FDIS	57/1853/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62351 series, published under the general title *Power systems management and associated information exchange – Data and communications security*, can be found on the IEC website.

In this standard, the following print types are used:

– ASN.1 notions is presented in bold Courier New typeface;

– when ASN.1 types and values are referenced in normal text, they are differentiated from normal text by presenting them in bold Courier New typeface.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

POWER SYSTEMS MANAGEMENT AND ASSOCIATED INFORMATION EXCHANGE – DATA AND COMMUNICATIONS SECURITY –

Part 9: Cyber security key management for power system equipment

1 Scope

This part of IEC 62351 specifies cryptographic key management, namely how to generate, distribute, revoke, and handle public-key certificates and cryptographic keys to protect digital data and its communication. Included in the scope is the handling of asymmetric keys (e.g. private keys and public-key certificates), as well as symmetric keys for groups (GDOI).

This part of IEC 62351 assumes that other standards have already chosen the type of keys and cryptography that will be utilized, since the cryptography algorithms and key materials chosen will be typically mandated by an organization's own local security policies and by the need to be compliant with other international standards. This document therefore specifies only the management techniques for these selected key and cryptography infrastructures. The objective is to define requirements and technologies to achieve interoperability of key management.

The purpose of this part of IEC 62351 is to guarantee interoperability among different vendors by specifying or limiting key management options to be used. This document assumes that the reader understands cryptography and PKI principles.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TS 62351-2, *Power systems management and associated information exchange – Data and communications security – Part 2: Glossary of terms*

ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), *Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9834-1:2012 | Rec. ITU-T X.660 (2011), *Information technology – Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree*

SCEP IETF Draft, *Simple Certificate Enrolment Protocol, draft-gutmann-scep-04.txt*

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*

RFC 5272, *Certificate Management over CMS (CMC)*

RFC 5934, *Trust Anchor Management Protocol (TAMP)*

RFC 6407, *The Group Domain of Interpretation*

RFC 6960, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*

RFC 7030, *Enrolment over Secure Transport*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TS 62351-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1
asymmetric keys
two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification

3.2
authorization and validation list
AVL
signed list containing information to an AVL entity about potential communications entities and possible restrictions on the communications with such entities

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.9]

3.3
authorization and validation list entity
AVL entity
entity, when acting as a relying party, which is dependent on an AVL issued by a designated authorizer

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.10]

3.4
authorizer
entity trusted by one or more entities operating as AVL entities to create, maintain and sign authorization and validation lists

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.11]

3.5
certification path
ordered list of one or more public-key certificates, starting with a public-key certificate signed by the trust anchor, and ending with the end-entity public-key certificate to be validated

Note 1 to entry: All intermediate public-key certificates, if any, are CA certificates in which the subject of the preceding public-key certificate is the issuer of the following public-key certificate.

[SOURCE: ISO/IEC 9594-8:2017, 3.5.18 | Rec. ITU-T X.509 (2016), 3.5.21]

3.6
certificate signing request
CSR
request issued when a new certificate or renewal of a certificate is required

Note 1 to entry: When the generated CSR is submitted to a CA, the CA signs the CSR using its private key and the CSR becomes the certificate.

[SOURCE: RFC 2986]

3.7
controllership
intersection of legal ownership, physical control, and logical control over a device or system, in which the nature of any contractual agreements between ownership and control of the device or system is not important in the context

3.8
cryptographic binding
use of one or more cryptographic techniques by a CKMS to establish a trusted association between a key and selected metadata elements

[SOURCE: NIST SP 800-130]

3.9
cryptographic key management system
CKMS
system for the management (e.g., generation, distribution, storage, backup, archive, recovery, use, revocation, and destruction) of cryptographic keys and their metadata

[SOURCE: NIST SP 800-130]

3.10
dataset
collection of data

3.11
digital signature
result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity, and signatory non-repudiation

[SOURCE: FIPS 186]

3.12
entity
generic term that covers human users, automation systems, software applications, communication nodes, field devices, and other types of assets

3.13
group controller/key server
GCKS
device that defines group policy and distributes keys for that policy

[SOURCE: RFC 3740]

3.14
group domain of interpretation
GDOI
domain that manages group security associations, which are used by IPsec and potentially other data security protocols

Note 1 to entry: These security associations protect one or more key-encrypting keys (KEK), traffic-encrypting keys (TEK), or data shared by group members. GDOI uses the notion of a group controller, which is used to support the establishment of security associations between members of a group.

[SOURCE: RFC 6407]

3.15
group member
GM
authorized member of a secure group, sending and/or receiving IP packets related to the group

3.16
hash function
(mathematical) function which maps data of arbitrary size into data of a fixed size called a digest

Note 1 to entry: Approved hash functions satisfy the following properties:

1) One-Way. It is computationally infeasible to find any input that maps to any pre-specified output.
2) Collision Resistant. It is computationally infeasible to find any two distinct inputs that map to the same output.

[SOURCE: ISO/IEC 9598-8:2017 | Rec. ITU-T X.509 (2016), 3.5.36]

3.17
hash message authentication code
HMAC
cryptographic code used for authentication with symmetric keys and for data integrity

[SOURCE: RFC 2104]

3.18
key distribution centre
KDC
centre which, in an IEC 62351-9 context, provides a network service that supplies temporary (symmetrical) session keys to predefined set of peers after successful authentication

Note 1 to entry: This is also known as Group Controller/Key Server (GCKS) (See GDOI).

3.19
message authentication code
MAC
cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modification of the data

[SOURCE: SP 800-63; FIPS 201]

3.20
object identifier
ordered list of primary integer values from the root of the international object identifier tree to a node, which unambiguously identifies that node

[SOURCE: ISO/IEC 9834-1:2012 | Rec. ITU-T X.660 (2011), 3.5.11]

3.21
online certificate status protocol
OCSP
protocol that enables applications to determine the (revocation) state of an identified certificate

Note 1 to entry: OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may be used to obtain additional status information. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response.

[SOURCE: RFC 6960]

3.22
pre-shared key
PSK
secret which is shared in advanced between the two entities, such as software applications or devices, to be able to authenticate themselves after establishing a secure connection

3.23
private key
(in a public-key cryptosystem) that key of an entity's key pair which is known only by that entity

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.49]

3.24
public-key certificate
public key of an entity, together with some other information, rendered unforgeable by digital signature with the private key of the CA which issued it

Note 1 to entry: A public-key certificate is often called an X.509 certificate or a digital certificate. However, such terms are ambiguous, as they could also mean attribute certificates, which are also defined by ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.57]

3.25
public-key cryptography standards
PKCS
specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the deployment of public-key cryptography

[SOURCE: www.rsa.com]

3.26
random number generation
RNG
process used to generate an unpredictable series of numbers

Note 1 to entry: Each individual value is called random if each of the values in the total population of values has an equal probability of being selected.

[SOURCE: NIST SP 800-57]

3.27
registration authority
those aspects of the responsibilities of a certification authority that are related to identification and authentication of the subject of a public-key certificate to be issued by that certification authority

Note 1 to entry: A registration authority may either be a separate entity or be an integrated part of the certification authority.

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T Rec. X.509 (2016), 3.5.60]

3.28
relying party
entity that relies on the data in a public-key certificate in making decisions

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.61]

3.29
secret key
cryptographic key, used with a secret key cryptographic algorithm that is uniquely associated with one or more entities and should not be made public

[SOURCE: FIPS 140-2]

3.30
security association
SA
relationship established between two or more entities to enable them to protect data they exchange

[SOURCE: NIST IR 7298 Rev.1]

3.31
security strength
ability of the security technologies to make it infeasible for a would-be attacker to bypass or subvert

Note 1 to entry: This is often measured in bits of security.

[SOURCE: NIST SP 800-130]

3.32
session key
in the context of symmetric encryption, key that is temporary or is used for a relatively short period of time

[SOURCE: RFC 2828]

3.33
simple certificate enrolment protocol
SCEP
protocol supporting the secure issuance of certificates to network devices in a scalable manner, using existing technologies whenever possible

Note 1 to entry: The protocol supports the following operations:

CA and RA public key distribution

Certificate enrolment

Certificate revocation

Certificate query

CRL query

[SOURCE: IETF Draft SCEP]

3.34
symmetric key
cryptographic key that is used to perform both the cryptographic operation and its inverse, for example to encrypt and decrypt, or to create a message authentication code and to verify the code

[SOURCE: NIST SP 800-63]

3.35
trust
firm belief in the reliability and truth of information or in the ability and disposition of an entity to act appropriately, within a specified context

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.2.1]

3.36
trust anchor
entity that is trusted by a relying party and used for validating public-key certificates in certification paths

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.71]

3.37
trust anchor information
at least the: distinguished name of the trust anchor, associated public key, algorithm identifier, public key parameters (if applicable), and any constraints on its use including a validity period

Note 1 to entry: The trust anchor information may be provided as a self-signed CA-certificate or as a normal CA-certificate (i.e., cross-certificate).

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.72]

3.38
trust anchor management protocol
TAMP
protocol used to manage a trust anchor store

[SOURCE: RFC 5934]

3.39
trust anchor store
trust anchor information collection at a relying party for one or more trust anchors

[SOURCE: ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), 3.5.73]

4 Abbreviations and acronyms

Additional abbreviations and acronyms are listed in IEC TS 62351-2.

ASN.1	Abstract Syntax Notation One
AVL	Authorization and Validation List
AVMP	Authorization and Validation Management Protocol
CA	Certification Authority
CASP	Certification Authority Subscription Protocol
CKMS	Cryptographic Key Management System

CIA	Confidentiality, Integrity, and Availability
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DOI	Domain of Interest
DSA	Digital Signature Algorithm
EST	Enrolment over Secure Transport
GCKS	Group Controller/Key Server
GDOI	Group Domain of Interpretation
GKDC	Group KDC
GM	Group Member
GMAC	Galois Message Authentication Code
GOOSE	Generic Object Oriented Substation Event
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
ID	Identity
IKE	Internet Key Exchange
ISAKMP	Internet Security Association and Key Management Protocol
KD	Key Download
KDC	Key Distribution Centre, a.k.a GKDC
KEK	Key Encryption Key
OCSP	Online Certificate Status Protocol
PDU	Protocol Data Unit
PEM	Privacy-enhanced Electronic Mail
PKCS	Public-Key Cryptography Standard
PKI	Public-Key Infrastructure
RA	Registration Authority
RNG	Random Number Generation
RSA	Rivest Shamir Adleman, public-key crypto system
SA	Security Association
SCEP	Simple Certificate Enrolment Protocol
SV	Sample Values
TAMP	Trust Anchor Management Protocol
TEK	Traffic Encryption Key

5 Cryptographic applications for power system implementations

5.1 Cryptography, cryptographic keys, and security objectives

The term cryptography refers to the use of a family of data transformation algorithms for the purposes of achieving specific security-oriented goals such as obscuring the contents of a message from unintended recipients (confidentiality), ensuring no one has tampered with message contents during transit (integrity), and verifying the claimed identity of the sender (authenticity). In order to provide confidentiality, integrity, and authenticity, modern implementations of ciphers use a cryptographic key as the changeable part of the data transformation, shifting the sensitivity of the data to the key rather than the data payload. This part of IEC 62351 is about management of these cryptographic keys to ensure security objectives are met in the context of power system implementations.

5.2 Types of cryptography

This part of IEC 62351 focuses on the management of keys for two basic types of cryptography: asymmetric and symmetric. Both of these types of cryptography rely on cipher suites and numerous well-accepted cryptographic algorithms. This document does not address the specification of cipher algorithms and configurations for particular implementations, but does address the unique requirements associated with management of both types of cryptographic keys for power system implementations.

For symmetric cryptography, the same key is used to encrypt and decrypt data, so this key must be known and be kept secret by both parties who are exchanging a symmetrically encrypted message. The different symmetric key algorithms are each identified by an object identifier (see 7.6).

Asymmetric cryptography uses a pair of mathematically related keys. One of the keys is the private key to be kept secret by the owner and the other one is the public key that may be publicly known. The public key is typically provided in a public-key certificate that binds the public key with the identity of the owner of the public-key certificate and thereby the corresponding private key. Such a public-key certificate is verified and digitally signed by a certification authority.

Asymmetric cryptography is primarily used for digital signature generation, in which the private key is used in combination with a hash algorithm to create a digital signature. The corresponding public key in combination with the same hash algorithm is used for validating the digital signature. Digital signatures provide data integrity (see 5.3.3), authentication (see 5.3.4), and non-repudiation (see 5.3.5). Asymmetric cryptography may also be used for key agreement whereby two or more entities can agree on a key in such a way that both influence the outcome, possibly using the Diffie Hellman technique ([1]). These different public-key algorithms are identified by an object identifier.

Certain public-key algorithms based on the RSA technology (see e.g., [25]), may be used for encryption and decryption. Data encrypted using the private key may be decrypted using the public key and vice versa.

While there are fundamental functional differences between RSA asymmetric cryptography and symmetric cryptography, neither type of cryptography is inherently stronger than the other. Their different characteristics cause them to be used for different purposes, and they may often complement each other. However, each has a distinct set of characteristics involving computational complexity and key length that create performance considerations for specific applications. Asymmetric cryptography used in power system implementations tends to be more computationally intense than comparably strong symmetric cryptography. These different performance characteristics thus guide choices for usage, although the specific algorithm, configuration, and platform choices may also greatly affect overall performance. For this reason, computational intensive RSA asymmetric key cryptography is often used just to exchange symmetric secret “session” keys, which can then be used to rapidly encrypt and decrypt all the messages during a “session”.

The robustness of a cryptographic algorithm is expressed as security strength measured in a number of bits that reflects the expected effort it takes to break the algorithm.

5.3 Uses of cryptography

5.3.1 Goals of cyber security

Cyber security is focused on countering key threats that includes the following functionality:

- Ensuring the source of commands and data is authenticated.
- Ensuring access to resources and data is authorized.
- Ensuring the integrity of data.
- Ensuring that system control and changes are authenticated and authorized.
- Preventing the disclosure of confidential data.
- Preventing non-repudiation (loss of accountability).
- Preventing denial of service (availability).

Traditionally, cyber security addresses confidentiality, integrity, and availability (CIA). However, for power system operations, the continued safe and reliable operation of the power system is the highest priority. This places the highest priority on the first three: data source authentication, access authorization, and integrity protection, while confidentiality and non-repudiation can be important for certain interactions.

Cryptography can help to address most of these threats, but primarily the first five threats – prevention of denial of service usually involves engineering designs and strategies. Therefore, for power system implementations, five primary goals are considered for the use of cryptography:

- Verifying the claimed identity of a message sender (authentication).
- Verifying that the sender has the right to access the requested data (authorization).
- Ensuring no one has tampered with a message during transit (integrity).
- Obscuring the contents of a message from unintended recipients (confidentiality).
- Associating specific actions with the entity that performed them (non-repudiation).

These cryptographic measures are discussed in 5.3.2 through 5.3.6. The ordering of these clauses does not imply priority for power system applications: in such environments, confidentiality usually has one of the lower priorities among these cryptographic goals. However, some of the cryptographic mechanisms to achieve these goals build upon each other, and so the goals are presented in the following order for ease of conceptual explanation.

5.3.2 Confidentiality

The cryptographic goal of data confidentiality in power system implementations is typically accomplished by encrypting the message using symmetric key cryptography. The message may be encrypted individually at the application level, at the underlying communications channel, or possibly both. As stated previously, the confidentiality of this message is dependent upon the sender and receiver maintaining the secrecy of the encryption key. Additionally, asymmetric cryptography is often used to negotiate and exchange symmetric session keys that are valid for a specified length of time, thus reducing the risk associated with keeping a specific session key secret.

5.3.3 Data integrity

Data integrity includes the detection of unauthorized changes to data. Any changes to the data as it transits the communication networks should be detected by the recipient. Cryptographic hash functions (e.g. SHA256) are used to detect any integrity violations of the received data.

The cryptographic goal of message integrity in power system implementations is typically accomplished by using another form of cryptographic algorithm called HMAC (keyed-hash message authentication code), a cryptographic hash function used in combination with a secret symmetrical cryptographic key. The result is a key hashing function, such as HMAC, that provides not only data integrity, but also validates the source of the data.

Alternatively, instead of symmetrical cryptographic, the sender may attach a digital signature to the message, providing additional cryptographic binding information on the data source (see 5.3.5).

5.3.4 Authentication

When entities, namely field devices, software applications, automation systems, and the users who interact with them, need to exchange data, it is critical that the source of this data is authenticated. To accomplish reliable authentication, data sources are provided with cryptographic keys that allow them to prove their identity to the recipients of the data. Public-key certificates and asymmetric cryptographic methods can be used to authenticate these entities as the sources of data.

5.3.5 Non-repudiation

Non-repudiation refers to the ability to bind an action (e.g., a command or a message) irrefutably to an issuing entity. It may bind a sender to the action of sending a specific message or a receiver to the action of receiving a specific message. This is accomplished using public-key certificates and asymmetric cryptography. A public-key certificate is a digitally signed statement that claims a public-private key pair is associated with a unique entity. Receivers of a signed message can prove that a message was sent by its originator since only the originator is presumably in possession of the private key needed to create the signature. If non-repudiation of receipt were required, the receiver would need to sign a copy of the received message and send it back to those requiring proof. Typically, a time-stamp is included in such signed messages since otherwise a signor could refute the message simply by declaring that their secret key is no longer secret.

5.3.6 Trust

One important part of securing digital communication is the need for trust. Entities (systems or devices) should only accept data (communicate) with entities that they can authenticate and trust (see 5.3.4). Public-key certificates provide the basis to establish such trust by asserting the association of a public-key certificate with a unique entity. Trust for a particular public-key certificate is established by validating a so-called certification path that has its root in a trust anchor that is trusted by the entity doing the validation, also called the relying party. A certification path is a chain of public-key certificates starting with the public-key certificate signed by the trust anchor ending with the public-key certificate to be validated. The concepts of relying party, trust anchor and certification paths are further described in Clause 6, 7.5 and 7.7 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

In some cases, trust may be established directly between a trust anchor and the relying party. However, a chain of trust can be established. If a relying party receives a public-key certificate from an entity with which it has a trust relationship, this public-key certificate may be validated without going through the complete certification path, instead relying on the chain of trust. In this case, the public-key certificate may not need to be issued and signed by the trust anchor, but may be signed by a certification authority that has established a chain of trust back to the trust anchor.

For power system implementations, certification authorities might be the relevant organizational unit within a company, the company itself, a governmental entity, or accepted third party.

Public-key digital certificates have a finite period of validity, after which they expire. Trust can also be revoked in the event of compromise. Management of public-key certificates is closely related to cryptographic key management, and is therefore covered in this standard. See 6.5.

6 Key management concepts and methods in power system operations

6.1 Key management system security policy

Cryptographic keys need to be protected. Therefore, every organization should develop a security policy for its key management system that establishes and specifies all of the requirements for protecting the confidentiality, integrity, availability, and source authentication of all cryptographic keys and metadata used by the organization. These protection requirements should cover the entire key life cycle, including when they are operational, stored, and transported. A key management security policy should also include the selection of all cryptographic mechanisms and protocols to be used throughout the organization's systems.

Key management systems also need security administrative support to ensure the security policies are followed and the procedures are maintained. The specification of this support is out-of-scope for this document, but can be found in other documents such as ISO/IEC 11770 ([8]) or in NIST SP 800-130 ([13]).

6.2 Key management design principles for power system operations

Cryptographic keys used with selected cryptographic functions can be used to secure the messages being sent between different entities, such as users, systems, software applications, communication nodes, and the potentially large numbers of equipment and devices that are often located at remote and often untrusted sites.

These cryptographic keys should be managed so that they can effectively and securely be provided to the entities that require secure exchanges of data. This key management must take into account many issues, ranging from the capabilities of entities, to the varied types of locations of these entities, to the timing for providing and revoking keys, and to protecting the key management processes themselves from attacks.

For instance, many smaller devices are limited in computational power and memory capacity, while the communication networks may also be limited in available bandwidth. Therefore, some of the key management techniques used in traditional enterprise information technology system environments with powerful systems and high bandwidth communications are not well suited to power system automation and communication environments.

To address these constraints, this document specifies different key management techniques that could be used for different requirements and constraints. Specifically it specifies how to manage keys for each of the cryptographic functions specified in the other IEC 62351 parts. Sources of guidance on key management design principles include:

- Reference [8] ISO/IEC 11770-1:2010 Information technology – Security techniques – Key management – Part 1: Framework, 6.1.2, D-3.4.
- Reference [10] ISO/IEC 11770-3:2008 Information technology – Security techniques – Key management – Part 3: Mechanisms Using Asymmetric Techniques.
- Reference [15] NIST 800-57, Part 1, 8.1.4 and 8.1.5.

6.3 Use of Transport Layer Security (TLS)

This part of IEC 62351 refers to Transport Layer Security as defined by RFC 5246. The details on how to use TLS in power systems are specified in IEC 62351-3.

6.4 Cryptographic key usages

Cryptographic keys are used for different purposes and in different phases of the product lifecycle and are applied as:

- Digital certificates and corresponding private keys: Used to authenticate entities when in a public-key infrastructure (PKI) environment.
- Pre-shared keys (e.g., for real-time communication): Used as a shared secret between entities to build up mutual authentication between them. This may be useful in non-PKI environments. Additionally, this scheme may be used during PKI enrolment, allowing an entity to authenticate itself against the certification authority (CA), for example when performing a certificate signing request (CSR). *Engineering tools could automate the generation of the entity registration password. Password complexity options are given by the organization security policies and CA supported possibilities.*
- Session secret keys (pair wise or group-based): Used for efficient encryption or integrity checks on communication messages.
- Session parameters (dedicated cryptographic algorithms): Used to support sessions (keys, lifetime, etc.).
- Cryptographic access tokens: Mostly used to transfer/provide authorization/access of resources to entities for a limited time.
- Other relevant entities or organizational credentials.

6.5 Trust using a public-key infrastructure (PKI)

6.5.1 Registration authorities (RA)

In PKI-based systems, entities are required first to prove their identity through a Registration Authority (RA). For humans, RAs may determine an individual identity through birth certificates, passports, or other official documents. For systems and devices, RAs may determine an entity's identity through a manufacturer-issued digital certificates or a unique one-time-password (OTP).

6.5.2 Certification authority (CA)

After its identity is established, a system or device needs to be bound to a certificate by a certification authority (CA). The manufacturer creates and digitally signs a certificate signing request (CSR) for the entity. The CA verifies the CSR and issues a digital certificate based on the data included in the CSR. CAs establish a trusted association between the entity's public key and the public-key certificate metadata by cryptographically binding the entity's public key and the metadata (i.e. the CA's digital signature computed on the combination of the public key and metadata). This digital certificate thus binds the entity's identity with its public key, so that there is now a trusted public/private key combination that can be used by the entity to exchange information with other entities. The trust in the entity's key thus relies on one's trust in the validity of the CA's key (see 6.5.3).

CAs may be operated either by an organization itself, allowing for a closed, organization-controlled communication group, or by a third party (service provider, system operator or grid manager) that is publicly accepted and hence has a wider scope of trust. Third-party CAs require a secure method for ensuring the valid identity of any new entity, which can range from in-person validation for humans to business-entity validation and to security chains of digital certificates linking previously validated digital certificates to new digital certificates.

6.5.3 Public-key certificates

A public-key certificate (a.k.a. digital certificate) is a digital document that binds the identity of entity to the public key private key pair of that entity. As noted in 6.5.2, this binding is verified by a digital signature by the issuing CA. In addition to the public-key and identity of the owner of public-key certificate, the public-key certificates holds verified information about validity period and the identity of the issuer. A public-key certificate may include extensions providing additional information. An extension is identified by an object identifier allocated by the organisation defining the extension. A public-key certificate may be issued to a CA and is then called a CA certificate or to an end entity and is then called an end-entity public-key certificate.

In a PKI environment, public-key certificates may be digitally signed by the CA of the organization to which the entity belongs. In some cases, multiple public-key certificates are created as an entity goes through a supply chain from manufacturer, to distributer, to purchaser, to installer, etc. Root certificates of the issuing CA become the trust anchors, if used. Typically, organizations should install only root certificates from CAs they trust, including public-key certificates issued by their own CA.

Public-key certificates and attribute certificates are defined by a base set plus extensions to the base set. Extensions are identified by an international register of object identifiers (OIDs). See 6.5.4.

OIDs are allocated by various international standards, including ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), and may be registered for private use. The OID namespace (root) value for IEC 62351 is 1.0.62351 and 1.2.840.10070. IEC TS 62351-8 utilizes the latter approach to define access tokens (authorizations). This part of IEC 62351 and every new upcoming part of IEC 62351 should use the first OID (1.0.62351).

6.5.4 Attribute certificates

Attribute certificates provide an effective way to separate the management of identity from the management of authorizations associated with an identity. Full separation can be achieved by managing public-key certificates with a PKI and attribute certificates with a "privilege management infrastructure" (which uses the same CA technology as a PKI).

As shown in Figure 1, attribute certificates can be used to extend the information in a public key certificate. They allow for instance for temporary enhancement of the permissions of the public key certificate holder by specific role-based access information. This approach has been leveraged in IEC TS 62351-8.

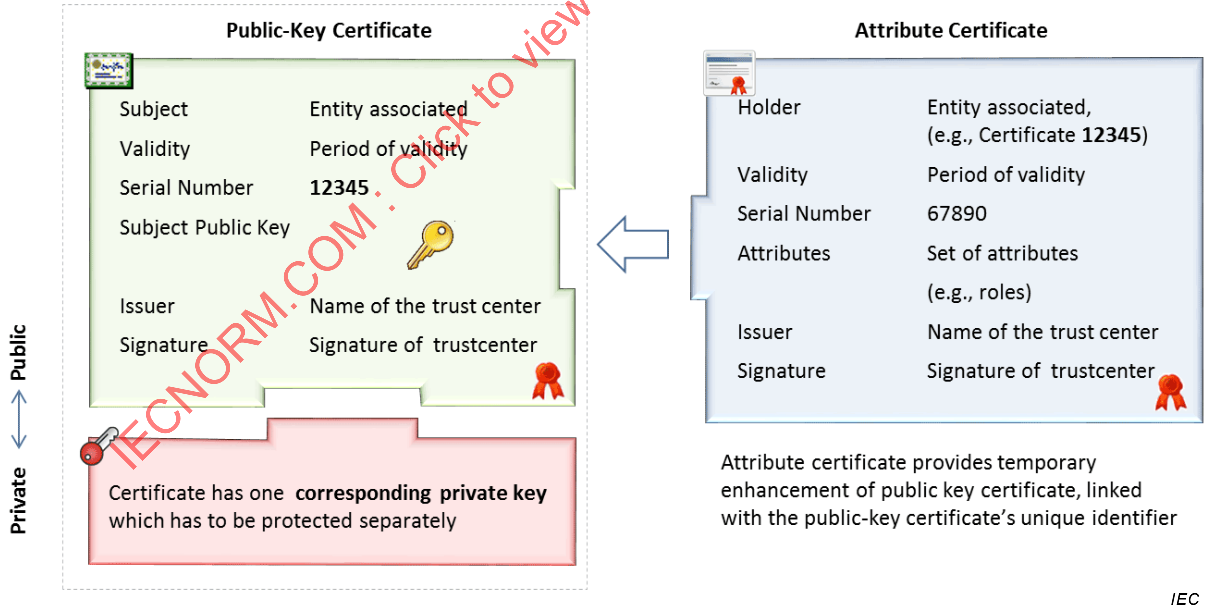


Figure 1 – Relationship between public-key certificates and attribute certificates

6.5.5 Public-key certificate and attribute certificate extensions

Both public-key certificates and attribute certificates allow extensions to be optionally included. Each such extension provides additional information.

An extension type consists of an object identifier (see 7.6) that identifies the type of extension and specifies the syntax for the associated information. In addition, it includes a Boolean that specifies whether a particular extension is flagged as critical or as non-critical. If an extension is flagged as critical, it cannot be ignored and the public-key or attribute certificate should be considered invalid if the relying party does not support the type of extension. If the extension is flagged as non-critical, the relying party may ignore the extension if the extension type is not supported. For more details, see 7.3 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016) specifies a number of general useable extensions. IEC TS 62351-8 specifies the `IECUserRoles` extension, which has been defined for power systems to provide means for role-based access control.

6.6 Trust via non-PKI self-signed certificates

Self-signed certificates play an important role in a PKI. Trust hierarchies or trust chains in PKIs are rooted by "root certificates" (see 6.5.3). PKI root certificates may be self-signed certificates known as trust anchors, created by trusted CAs that allow this trust to be distributed.

Under certain circumstances, non-PKI self-signed certificates may be more easily implemented than the more complex PKI signed certificates in order to establish trust. This may especially be the case in smaller deployments with only a limited number of entities. Self-signed certificates are public/private key pairs internally generated by an entity, where the private key is used to sign the entity's own public key together with additional information, such as the subject name, validity time, etc. So that they can be used in the same way as PKI based certificates, e.g., to authenticate interactions in TLS connections, self-signed public-key certificates are required to comply with the public-key certificate structure as described in ISO/IEC 9594-8.

Self-signed certificates cannot be generally accepted as trustworthy. Hence, to enable the acceptance of self-signed certificates by a limited group of entities for authentication, they should only be used in conjunction with authorization and validation lists (see 6.7).

The level of complexity of using entity specific self-signed certificates and authorization and validation lists can be compared with the use of pre-shared keys. Self-signed certificates are issued per entity while pre-shared keys are issued per paired connection. Thus while the number of self-signed certificates will depend upon the number of end points, the number of pre-shared keys will depend upon the number of connections between the end points.

As self-signed certificates need to be public-key certificates, their application could open the migration path to PKI-based public-key certificate implementations, so that self-signed certificates may easily be replaced with PKI-based certificates once those are available.

6.7 Authorization and validation lists

6.7.1 General

Authorization and validation lists (AVLs), also known as White Lists, provide information about potential communications entities and possible restrictions on the communications with such entities in a particular environment. They are specified in Clause 11 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

An AVL is used by an entity when that entity is acting as a relying party. Such an entity is called an AVL entity. An AVL is placed in an entity called authorizer that is responsible for the content of the AVL, i.e., the authorizer is trusted by the AVL entity to provide valid information. An AVL is signed by the issuing authorizer.

The communication between the authorizer and the AVL entity has to be protected as any other communication with respect to integrity, authenticity and possible confidentiality. To simplify validation of the AVLs, it is recommended that an authorizer be closely related to the AVL entities it serves, e.g. being within the same organization so that a trust relationship may be established between the authorizer and the AVL entities it serves (see 5.3.6).

AVLs may be used either in non-constrained environments or in constrained environments as detailed in 6.7.2 and 6.7.3.

6.7.2 AVLs in non-constrained environments

A non-constrained environment is an environment where AVL entities are capable of performing traditional validation of received public-certificates.

AVLs are the used to restrict communication within a specified set of other entities, and may impose restrictions on such entities beyond path length, policy and name restriction as detailed in ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016). 8.3.4 specifies such additional restrictions imposed by AVLs.

The entities with which communication is possible are identified in the AVL either by reference to their public-key certificates or by the name structure of the organization to which the entity belong.

6.7.3 AVLs in constrained environments

An entity may be constrained in many ways:

a) It is battery driven and needs to limit processing to save on the battery.
b) It has little processing capabilities.
c) Its communication channel has limited bandwidth.
d) It has limited storage.
e) It may have stringent time constraints having to respond to incident very quickly not allowing time for external communication to validate received public-key certificates.

In addition to what is described in 6.7.2, the authorizer is responsible for validating that all public-key certificates represented in the AVL are valid and may be trusted at all times. It is also the responsibility of the authorizer to be updated on the validity of the public-key certificates in their AVLs. The authorizer also makes the judgement as to when an expired or revoked public-key certificate should or should not be used in cases where its withdrawal might affect a critical entity. An AVL entity assumes that a listed public-key certificate is valid to use.

This mode of operation requires communication between the authorizer and the CAs that issued the public-key certificates listed in the AVL.

6.7.4 Use of self-signed public-key certificates in AVLs

Use of self-signed public-key certificates is not recognized by ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), but is provided here as an added capability.

By placing the self-signed public-key certificates in an AVL, the authorizer takes the responsibility to ensure that these self-signed public-key certificates may be safely used by digitally signing the AVL. How the authorizer maintains information about the validity of the self-signed public-key certificates is outside the scope of this part of IEC 62351.

6.8 Trust via pre-shared keys

Under special conditions, pre-shared keys may be manually installed in entities. These pre-shared keys, which are symmetric keys shared in advance among communicating entities, may be used to establish trust for peer-to-peer interactions and may also be used to secure further key management steps. The pre-shared keys, which need to be kept secret, need to be installed by a trusted entity using trusted installation methods. Guidance on the installation of pre-shared keys is provided in NIST SP 800-133 ([14]).

6.9 Session keys

Session keys are used for a relatively short time, ostensibly for a "session" between two entities, but in reality to ensure that cryptographic keys are replaced in a timely manner. That update time period is generally related to the cryptographic strength of the keys and the time it might take an attacker to break the key.

For instance, session keys are often used in conjunction with secret keys with entities that are compute-constrained and/or communications-constrained. Asymmetric cryptography, which can be computationally intensive, may be used periodically to update the session keys in the entities. These session keys can then be used by these entities for a few hours or even a few days. Usually a short overlap time is allowed between the expiration of a previous session key and the provision of the new session key, in order to ensure that communications can be maintained continuously (without doubling computational load).

6.10 Protocols used in trust establishment

6.10.1 Certification request

The certification request is specified as PKCS#10 in RFC 2986 ([20]). A certification request is sent from an entity to the registration authority (RA), which may be co-located with the CA or be physically separated.

A certification request contains the information to be included in the **subject** and **subjectPublicKeyInfo** components of the public-key certificate to be generated. In addition, some directory attributes may be included.

RFC 2985 defines two attribute types, one providing a challenge password and another one providing a list of extensions to be included in the generated public-key certificate.

6.10.2 Trust Anchor Management Protocol (TAMP)

The Trust Anchor Management Protocol (TAMP), as defined in RFC 5934, specifies a protocol for the management of trust anchors stores (TAs) and community identifiers stored in a trust anchor store. The protocol makes use of the Cryptographic Message Syntax (CMS), which provides integrity protection and data origin authentication. The protocol can be used to manage trust anchor information within trust anchors stores. Trust anchor information may be represented as a CA-certificate, as an unsigned CA certificate or as a **TrustAnchorInfo** object as defined by RFC 5914.

6.10.3 Simple Certificate Enrolment Protocol (SCEP)

The Simple Certificate Enrolment Protocol (SCEP) was developed to simplify the enrolment of large numbers of devices and to make the issuing and revocation of digital certificates as scalable as possible. Entities can use SCEP to request their digital certificate electronically using the PKI certificate forms PKCS#7 and PKCS#10 over HTTP. The key material is generated only on the client side. SCEP is defined by the IETF in draft-nourse-scep-23. Based on the dependency on PKCS#7, SCEP targets the enrolment in RSA based infrastructures. SCEP may be terminated at the RA or at the CA directly.

NOTE 1 Draft-nourse-scep-23 is a historic draft, which itself recommends using either CMP (RFC 4210, see 6.10.4) or CMC (RFC 5272, see 6.10.5) if SCEP support is not required by the infrastructure.

NOTE 2 Meanwhile there exists an attempt to bring SCEP to RFC status. The draft-gutmann-scep-04 targets the improvement of SCEP by addressing recently discovered issues. Nevertheless, it will continue to be bound to the RSA environment.

6.10.4 Internet X.509 PKI Certificate Management Protocol (CMP)

The Certificate Management Protocol (CMP) is an Internet protocol used for obtaining public-key certificates in a PKI. It defines a protocol for the interaction between a client and the PKI components. It provides more options compared to SCEP but is also more complex. Besides CRL retrieval, certification request handling, identification/authorization option for the requester as well as proof of possession of the associated private key, CMP provides additional functionality like cross certification and certificate revocation, which is mandatory to be supported. CMP supports the client and server side generation of key material.

CMP is defined in RFC 4210 ([30]). The transport of CMP itself is defined in a separate document, RFC 6712 ([46]).

6.10.5 Certificate Management over CMS (CMC)

Like SCEP, and in contrast to CMP, certificate management over Cryptographic Message Syntax (CMC) enhances PKCS#7 and utilizes PKCS#10. It provides more mandatory options compared to SCEP and results therefore in greater complexity. Besides CRL retrieval, certification request handling, identification/authorization option for the requester as well as proof of possession of the associated private key, CMC provides additional functionality like cross certification and certificate revocation. CMC defines a simple and a full PKI request/response handshake, but requires both to be implemented. CMC supports the client and server side generation of key material.

CMC is defined in RFC 5272. The transport of CMC itself is defined in a separate document, RFC 5273 ([35]).

6.10.6 Enrolment over Secure Transport (EST)

Enrolment over Secure Transport (EST) is based on CMC and defines some of the CMC functionality as optional, resulting in reduced complexity of the protocol. It may be seen as profile of CMC. In EST, only the simple PKI request/response interaction is mandatory, while the full procedure support is optional. From a functionality perspective, EST can be seen as an evolvement of SCEP. EST utilizes TLS as a secure channel and leverages the authentication of the TLS channel for identification and authorization of the requester by binding the CSR to the actual TLS session. Besides the proof of identity, the CMC portion provides proof-of-possession of the private key corresponding to the public key in the CSR.

Additionally to the certificate enrolment and management functionalities, EST allows for the management of trust anchors on devices. This allows for the exchange of CA certificates (trust anchors) and corresponding trust chains. Moreover, it supports certificate attribute retrievals from a client side to query additional information or boundary conditions prior to generating a CSR. EST may be terminated at the RA or at the CA directly. Terminating EST at the RA leaves the communication between the RA and the CA untouched.

EST is defined in RFC 7030 ([47]).

6.10.7 Summary view on the different protocols

In summary:

- SCEP is good for making use of an existing PKI infrastructure and for legacy support of RSA based public-key certificates.
- EST is gaining popularity over the near term and copes with requirements for supporting both, RSA and ECC based public-key certificates.
- TAMP provides more flexibility for trust anchor management than the base functionality provided in some of the enrolment protocols.

6.11 Group keys

6.11.1 Purpose of group keys

For peer-to-peer or multicast interactions between entities that have stringent performance requirements, group key management is more efficient than pair-wise key management. Group key management typically uses a combination of asymmetric and symmetric cryptography.

Such peer-to-peer and multicast interactions may require the use of group-based Traffic Encryption Key (TEK) to provide authentication or confidentiality of the data exchanged. To realize the setup of a group-based key, one system or device typically is designated as the group controller, which in turn authenticates other entities via their certificates or pre-shared keys. After successful authentication of the entities, the group controller distributes the actual group key to them. Hence, the group controller resembles the functionality of a Key Distribution Centre (KDC) (see Figure 2).

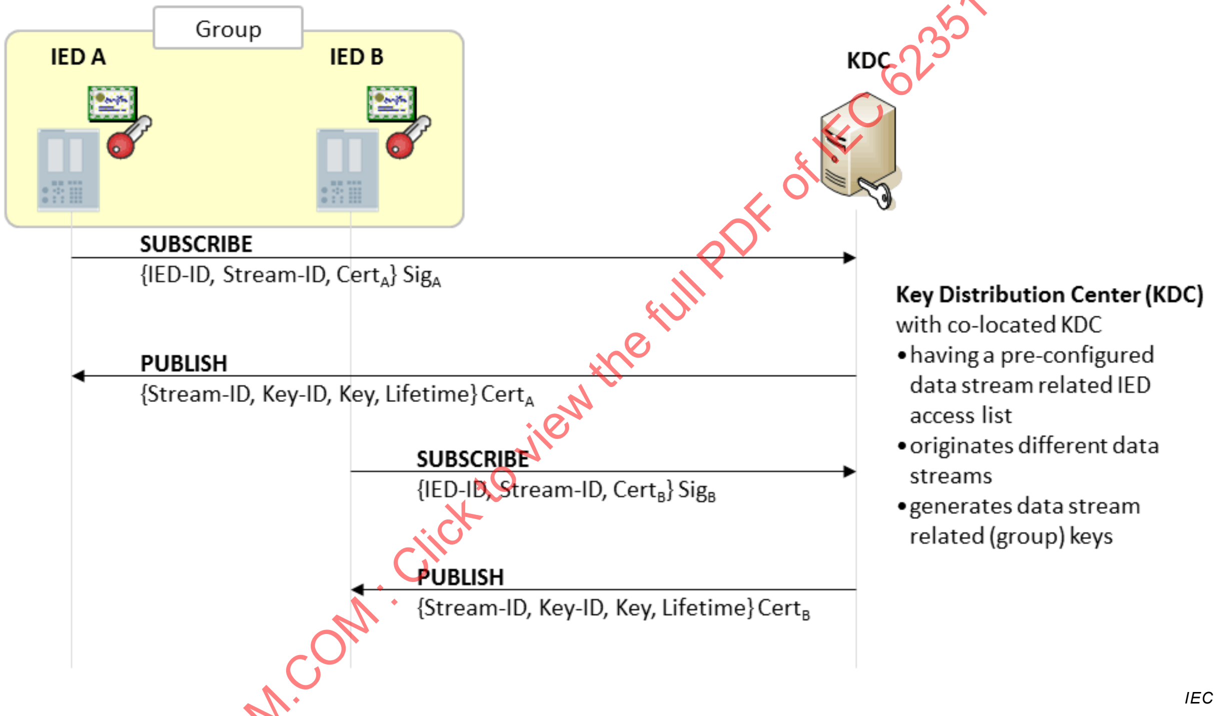

Group
IED A
IED B
KDC
SUBSCRIBE
{IED-ID, Stream-ID, Cert$_A$} Sig$_A$
PUBLISH
{Stream-ID, Key-ID, Key, Lifetime} Cert$_A$
SUBSCRIBE
{IED-ID, Stream-ID, Cert$_B$} Sig$_B$
PUBLISH
{Stream-ID, Key-ID, Key, Lifetime} Cert$_B$
Key Distribution Center (KDC)
with co-located KDC
• having a pre-configured data stream related IED access list
• originates different data streams
• generates data stream related (group) keys
IEC

Figure 2 – Group key management distribution

Figure 2 shows the subscription of an Intelligent Electronic Device (IED) for a data stream. The data stream is associated with a communication group. Note that the group controller may be a separate entity or may be deployed within any IED.

Note that there exist different protocols for distributing group keys. Group Domain of Interpretation (GDOI) has been selected as the most appropriate protocol for the power system automation.

6.11.2 Group Domain of Interpretation (GDOI)

6.11.2.1 General

The Group Domain Of Interpretation (GDOI) method defined in RFC 6407 supports the distribution of symmetric group keys (i.e. TEK) to all pre-configured or otherwise enrolled entities, typically devices. This method requires a Key Distribution Centre (KDC) that is responsible for distributing symmetric session key sets to enrolled entities following the GDOI process ([44]). This process uses point-to-point communications between the KDC and each group member (GM) to distribute the symmetric keys. The group key itself is then applied to protect subsequent multicast communications within the group.

Note that a KDC failure will disrupt group communication, so KDC redundancy is imperative. However, the method for achieving KDC redundancy is outside the scope of this document at this point.

6.11.2.2 GDOI Phase 1: Internet Key Exchange (IKE) Phase 1

The GDOI Phase 1 security association provides mutual authentication and authorization. It is used by the protocol participants to execute a Phase 2 exchange. The GDOI RFC incorporates (i.e., uses but does not redefine) the Phase 1 security association definition from the Internet DOI [RFC2407], [RFC2409].

NOTE Although RFCs 2407, 2408, and 2409 were obsoleted by [RFC4306] (and subsequently [RFC5996]), they are used by GDOI because the protocol definitions remain relevant for ISAKMP protocols other than IKEv2.

6.11.2.3 GDOI Phase 2: GDOI symmetric key distribution

As defined in the GDOI process, GDOI performs IKEv1 Phase 1 to provide a secure method of mutual authentication between the KDC and each of the group members (see Figure 3). The symmetric key transfer from the KDC to the entities is initiated after mutual authentication.

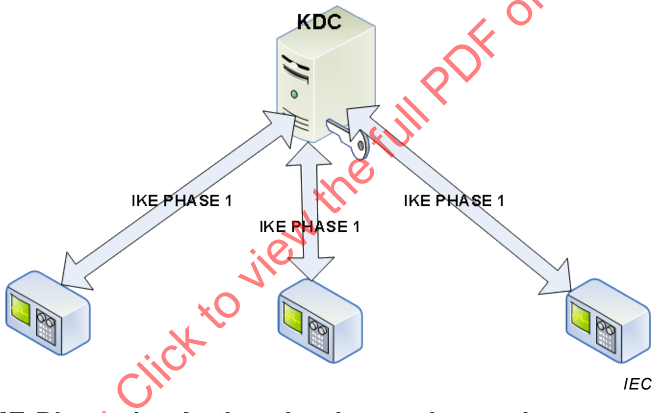


Figure 3 – GDOI IKE Phase 1 – Authentication and securing communication channel

In the GDOI process ([44]), two methods are defined on how to transfer the keys from the KDC into the entities. The first method is the Pull Method (keys are pulled by the entities from the KDC) and the second is the Push Method (the keys are pushed from the KDC into the entities) model.

Either a GROUPKEY-PULL exchange or GROUPKEY-PUSH exchange may be used to distribute symmetric keys to the entities. GROUPKEY-PULL is required, as it is the only method that works in conjunction with the Phase 1 authentication and authorization. There are advantages associated with each of the two methods. GROUPKEY-PULL offers control over traffic delivery while GROUPKEY-PUSH allows for a timely revocation or eviction of an entity and better efficiency.

GDOI has been extended to support IEC 62351 Security Services that are described in 6.11.2.6.

6.11.2.4 GROUPKEY-PULL registration protocol exchange

The GDOI Pull Method is illustrated in Figure 4.

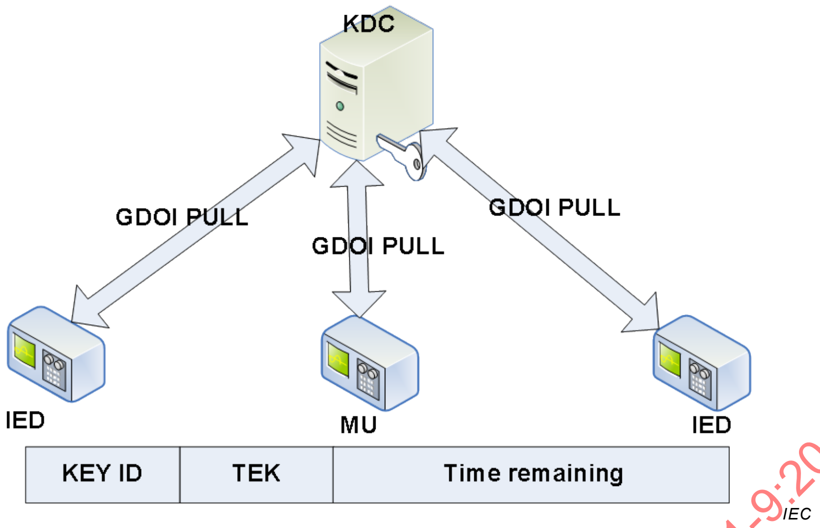


Figure 4 – GDOI Pull Phase 2

There are two phases of communication in the GROUPKEY-PULL method:

- Phase 1: Connection establishment and authentication of the two participating entities, a group member and KDC.
- Phase 2: Determining the policies in use for the requesting group and downloading the keys: This is accomplished through a Group Member (GM) making a GDOI Identification Payload Request to the KDC. The KDC responds to this request with the policies (e.g. encryption and signature algorithms) that are supported. The policies are returned using GDOI SA, which returns a SA TEK payload. If the GM does not support the policies, the communications should abort. If the GM can support the policies, it issues an acknowledgement and the KDC replies with the Key Download (KD) payload.

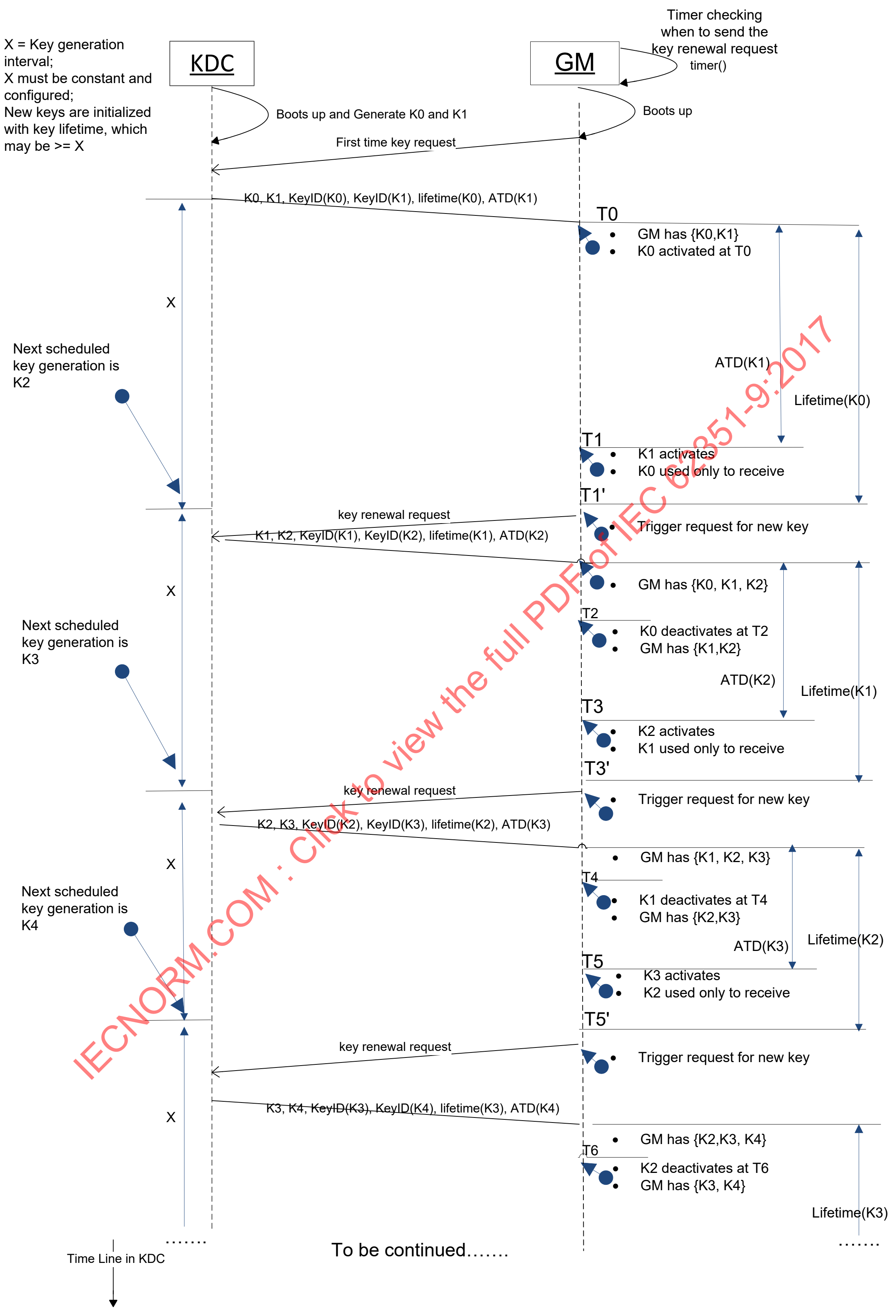
The renewal of the session key is normally triggered before the 'Remaining Lifetime Value' of the current key has expired. "Remaining Lifetime Value" (a.k.a lifetime) is one of the parameters returned in the SA TEK payload and it signifies the number of seconds remaining before the SA it is associated with expires.

This process is shown in Figure 5 as the following sequence:

- The KDC creates a session key for a group of devices along with its 'Key Identifier (KeyID)'[1], a Remaining Lifetime Value and SA Time Activation Delay (SA_ATD) parameter. It maintains that session key for the group throughout that session:
  - 'KeyID': is the unique identifier for the session key sent by the KDC in the GDOI GROUPKEY-PULL or GROUPKEY-PUSH message payload.
  - Remaining Lifetime Value: is the number of seconds remaining before the SA it is associated with expires.
  - 'SA Time Activation Delay (SA_ATD)': This parameter specifies when the key is expected to be used because the KDC will sometimes distribute an SA TEK in advance.
- Upon initialization, a GM sends a request to the KDC for the appropriate group session key.
- The KDC sends two SAs and associated keys, one (K0) that is current with SA_ATD zero, so it can be used right away, and a second one (K1) with an SA_ATD non-zero, to be used later on. KDC synchronizes the two SAs, so that the time delay SA_ATD for the second key is less than the lifetime of the first key.

[1] GDOI refers to this as an SPI.

- When the GM obtains the two session keys K0 and K1, it starts using K0 right away and starts (two) timers, one for K0 remaining lifetime and an SA_ATD timer for when K1 will be activated.
- GM switches to the new key K1 when the SA_ATD time expires.
- When K0 Remaining Lifetime Value expires, the GM sends a key renewal request to the KDC to obtain the next session key K2. Note that the KDC should always keep at least two SAs, a current active one, and a next with its ATD set as non-zero. If the trigger for creating a new SA is based on the expiration of the current one, then the KDC should have no more than two SAs at any time. However, if the trigger to create a new SA is based on the ATD expiration of the next SA, there will be situations where the KDC must hold three SAs for a group: a current, a next and a new one. The KDC may send all these SAs during the pull, so a GM is prepared to receive them. However, if the GM sends the pull request at the expiration of the current key, and not at the expiration of ATD for the next key, it always receives only two SAs and keys.


X = Key generation interval;
X must be constant and configured;
New keys are initialized with key lifetime, which may be >= X
KDC
GM
Timer checking when to send the key renewal request
timer()
Boots up and Generate K0 and K1
Boots up
First time key request
K0, K1, KeyID(K0), KeyID(K1), lifetime(K0), ATD(K1)
T0
GM has {K0,K1}
K0 activated at T0
X
Next scheduled key generation is K2
ATD(K1)
Lifetime(K0)
T1
K1 activates
K0 used only to receive
T1'
key renewal request
Trigger request for new key
K1, K2, KeyID(K1), KeyID(K2), lifetime(K1), ATD(K2)
GM has {K0, K1, K2}
X
T2
K0 deactivates at T2
GM has {K1,K2}
Next scheduled key generation is K3
ATD(K2)
Lifetime(K1)
T3
K2 activates
K1 used only to receive
T3'
key renewal request
Trigger request for new key
K2, K3, KeyID(K2), KeyID(K3), lifetime(K2), ATD(K3)
GM has {K1, K2, K3}
X
T4
K1 deactivates at T4
GM has {K2,K3}
Next scheduled key generation is K4
ATD(K3)
Lifetime(K2)
T5
K3 activates
K2 used only to receive
T5'
key renewal request
Trigger request for new key
K3, K4, KeyID(K3), KeyID(K4), lifetime(K3), ATD(K4)
X
GM has {K2,K3, K4}
T6
K2 deactivates at T6
GM has {K3, K4}
Lifetime(K3)
…….
…….
Time Line in KDC
To be continued…….

K0,K1,…. Kn denote TEKs
T0,T1,…. Tn represent time points when new TEKs are received or activated
T2,T4 … represent time point when expired TEKs are deactivated.

IEC

Figure 5 – Key renewal triggered by the entities

6.11.2.5 GROUPKEY-PUSH rekey protocol exchange

The KDC policy may include the use of the “push” method for rekeying, in which case a datagram initiated ("pushed") by the KDC is delivered to all group members using a IP multicast address or is sent to a specific GM using IP unicast. The GROUPKEY-PUSH method is useful for evicting group members that may have been compromised and may have had their certificates revoked. Using this method the KDC can rekey all authorized group members while excluding the group member that is being evicted.

6.11.2.6 GDOI protocol support for IEC 62351 security services

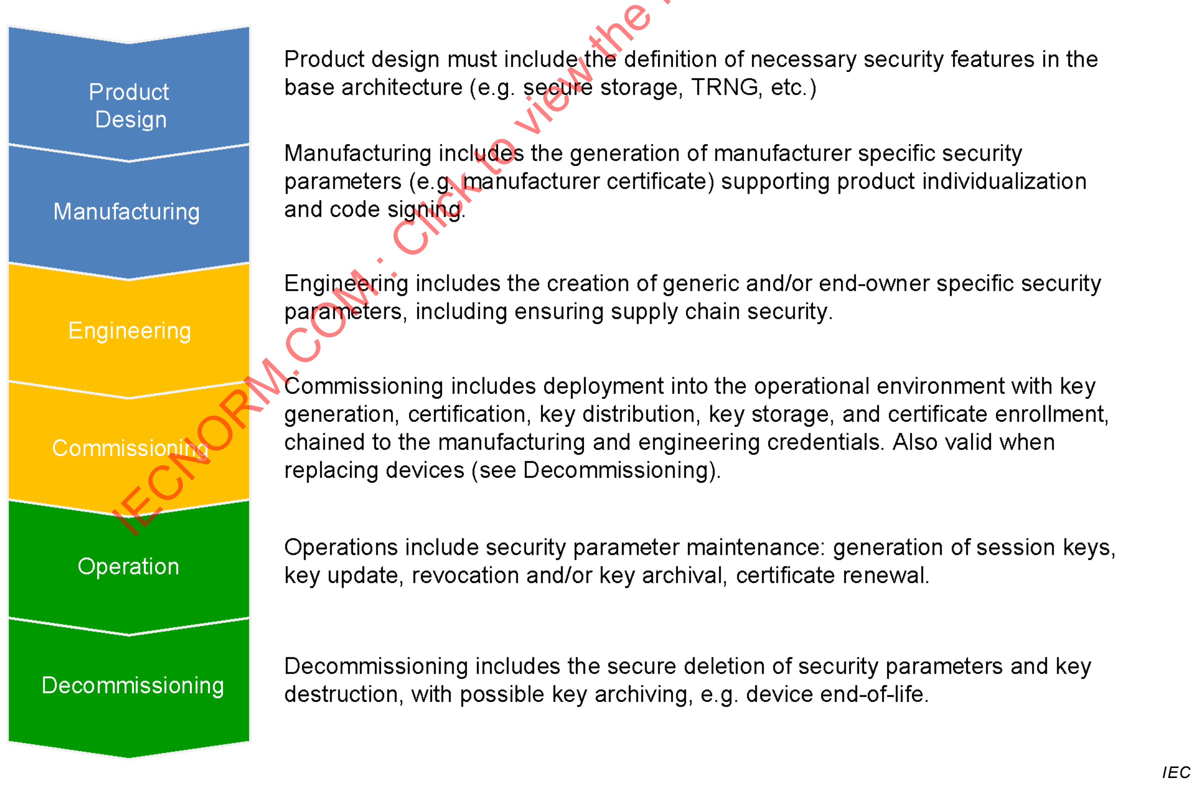
The Internet Draft for “GDOI Protocol Support for IEC 62351 Security Services” was developed to address the use of GDOI for the IEC 61850 power utility automation family of standards. It describes the methods to be used to distribute security transforms that are used for the protection of messages for some IEC 61850 security protocols.

The protection of the messages is defined within IEC TS 62351-6 [IEC-62351-6], IEC 61850-8-1 [IEC-61850-8-1], and IEC 61850-9-2 [IEC-61850-9-2]. Protected IEC 61850 messages typically include the output of a Message Authentication Code (MAC) and may be encrypted using a symmetric cipher such as the Advanced Encryption Standard (AES).

6.12 Key management lifecycle

6.12.1 Key management in the life cycle of an entity

Key management is part of the life cycle of entities (devices and systems), generally following the flow in Figure 6. Engineering and commissioning should be done securely.



Product design must include the definition of necessary security features in the base architecture (e.g. secure storage, TRNG, etc.)

Manufacturing includes the generation of manufacturer specific security parameters (e.g. manufacturer certificate) supporting product individualization and code signing.

Engineering includes the creation of generic and/or end-owner specific security parameters, including ensuring supply chain security.

Commissioning includes deployment into the operational environment with key generation, certification, key distribution, key storage, and certificate enrollment, chained to the manufacturing and engineering credentials. Also valid when replacing devices (see Decommissioning).

Operations include security parameter maintenance: generation of session keys, key update, revocation and/or key archival, certificate renewal.

Decommissioning includes the secure deletion of security parameters and key destruction, with possible key archiving, e.g. device end-of-life.

Figure 6 – Key management in product life cycle

As illustrated in Figure 7, manufacturers need to start the key management life cycle for an entity by providing a security manifest of identity information, such as a manufacturer unique identifier, or one-time-password, or certificate from a CA, or manufacturer’s entity certificate for each of their devices and systems. At each change in ownership or even status (e.g.

warehoused vs. deployed), new certificates should be registered, providing a trusted supply chain of product identity. When these entities are eventually deployed, their operational certificates are used to enrol the entity in operations so that they can be authenticated and commence their information exchanges. Shortly before these operational certificates are due to expire, they should be renewed.

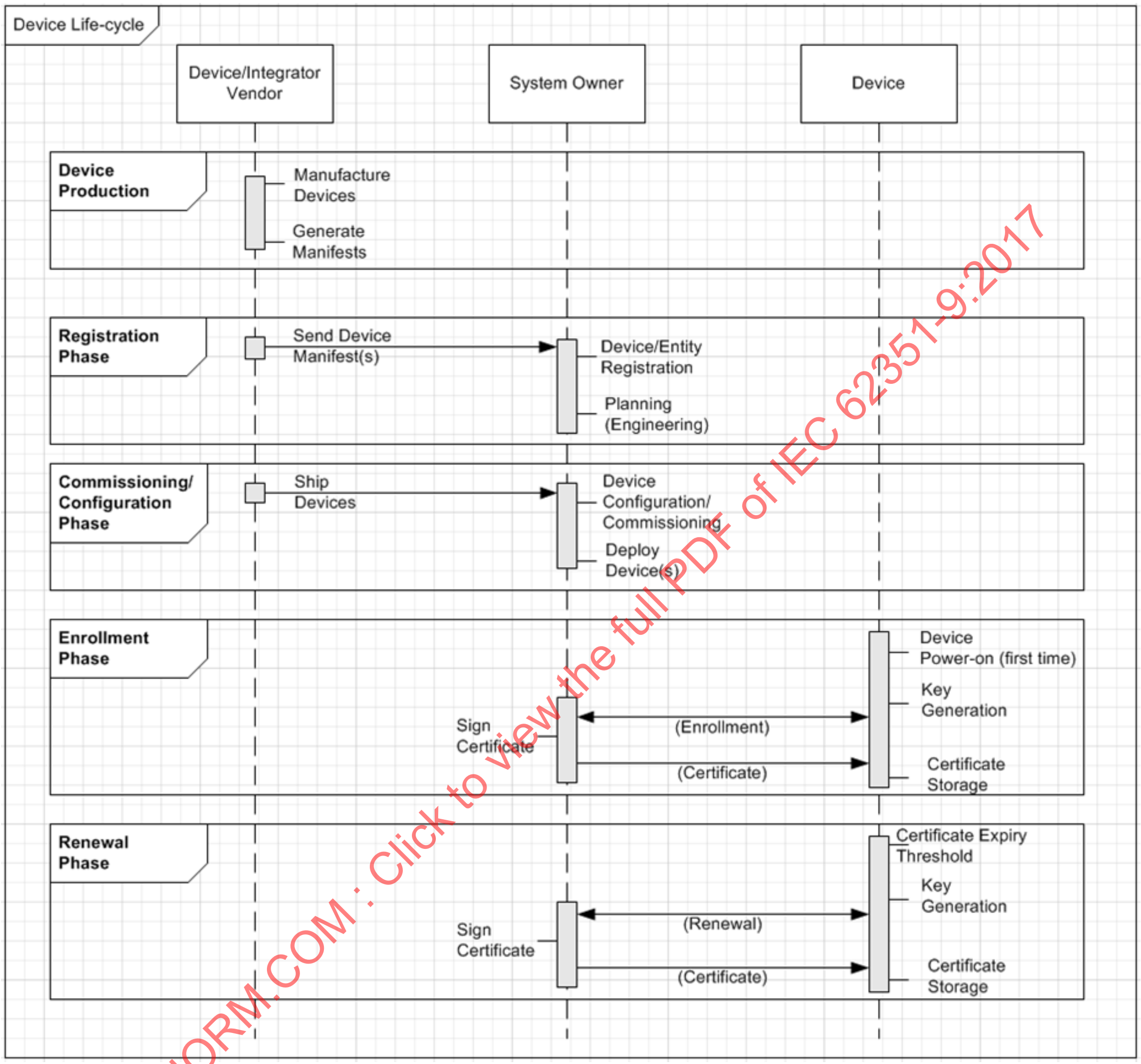


Figure 7 – Simplified certificate life cycle

6.12.2 Cryptographic key lifecycle

Cryptographic keys have a life cycle within the overall certificate lifecycle. They are created, distributed, installed, applied, updated, and destroyed to meet the key management security policy requirements. As an example, the typical life cycle of keys found in PKI based systems is depicted in Figure 8. The general key life cycle management is discussed in more detail in ISO/IEC 11770 (cf. [8]) and NIST SP 800-130 [13]. The life cycle of certificate management, which is part of key management, is depicted here on an abstract level. More details are found in the sub clauses of 6.13 addressing certificate management.

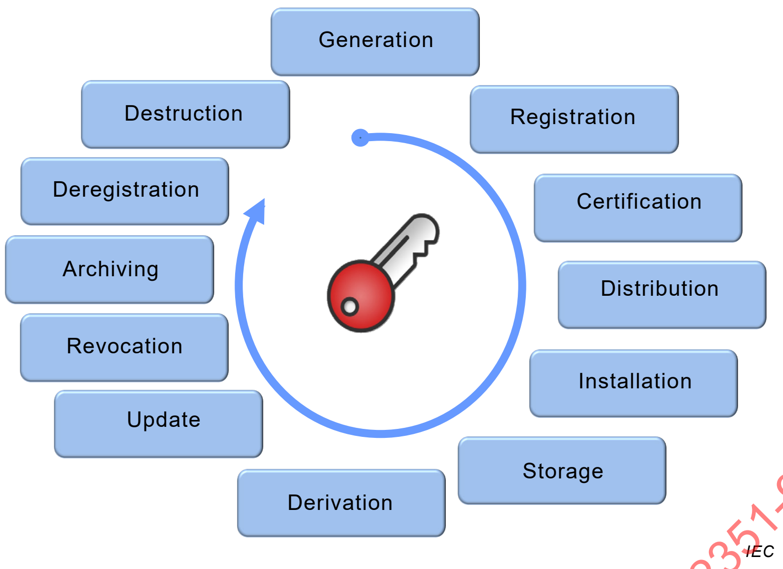


Figure 8 – Cryptographic key life cycle

The following list briefly describes each of the possible stages in the life cycle of a cryptographic key. All stages are required for asymmetric keys, while registration, certification, deregistration and revocation are not required for symmetric keys:

- Generation: Cryptographic keys can be created by the entity itself if it has the appropriate capabilities. Alternatively, keys may be created externally and installed on the target entity. Often this latter approach is used if the entity cannot support one of the critical components in generating keys, namely the random number generator (RNG) which must ensure randomness to a high degree (see Annex B). Keys might also need to be created externally if they must be archived (for those keys needing to be archived, e.g. for encryption keys used for stored data). When certificates are about to expire, entities need to apply for certificate renewal.
- Registration: If certificates and the PKI process are used, registration through a Registration Authority (RA) establishes the “identity” of an entity. The RA then provides a certificate request to the Certification Authority (CA).
- Certification: After registration of an entity by an RA, the CA provides a digitally signed certificate to the entity that thus connects its public key with the private key held by the entity. Depending on the key generation, this can be part of the key generation in a trust centre or may be done using information sent in a certificate signing request (CSR).
- Distribution: Key distribution comprises the process to transport keys as well as key management information to authorized entities in a secure manner. Private (secret) keys, although ideally generated in end devices and thus not requiring distribution, must be distributed using secure means. Public keys, within public-key certificates, can be distributed using non-secure means since end devices can verify their authenticity directly by verifying the certificate signature.
- Installation: A key may be installed in an entity during the manufacturing and/or engineering processes (pre-shared key), or may be installed later via on-line procedures. The latter process requires communication with a security server, out of band using a separate communication channel, or in-band as part of a service communication.
- Storage: The private/secret keys should be securely stored in the entity. Secured key storage should be compliant with FIPS 140-2 or ISO/IEC 19790.
- Derivation: Cryptographic keys used as session keys or other short-term keys can be derived from the private or secret keys that are stored in the entity.
- Update: Cryptographic keys need to be regularly updated. Cryptographic keys have a dedicated lifetime, e.g., user certificates typically have a lifetime of 2 years, while server

certificates are typically limited to 1 year, pre-shared keys to a few weeks or months, and session keys to a few hours and/or a few thousand packets.

- Revocation: Key revocation may occur when a key is no longer authorized to be used. This may be the case if the person leaves the organization, if a device is removed from service or changes its role, or if a private key is compromised or suspected of being compromised.
- Archiving: Secret keys needed to decrypt long-term stored data should be archived in secure confidential storage to enable data recovery (to allow access to store encrypted data later). In addition, to support later signature verification, public-key certificates should also be archived, although secure confidential storage is not needed.
- De-Registration: This procedure is typically provided by the registration authority to remove the association of an entity with a dedicated key. It is typically used in the key destruction phase.
- Destruction: When a cryptographic key is destroyed, it cannot be recovered or used. This occurs at the end of the cryptographic key life cycle.

6.13 Certificate management processes

6.13.1 Certificate management process

The certificate management process may be repeated upon certificate creation, each change of ownership or use of the entity, and when the certificate is about to expire.

6.13.2 Initial certificate creation

Just as a birth certificate is the ultimate proof of the identity of a person and the physical presence of that person at a registration office is required for personal identification, a device or system needs to have ultimate proof of its identity, usually when it is still on the manufacturer’s floor and when received by the new owner. This is achieved by registering it with a RA. Registration could be manual for individual entities or could be handled in bulk for large numbers of entities. The registration process then acts as the source of the supply chain trust or provenance of the entity.

6.13.3 Enrolment of an entity

Enrolment is the process by which the entity receives a signed certificate from the CA. Some different enrolment scenarios include the following entity conditions:

- One-time unique password without certificate.
- With CA-issued certificate.
- With known (white listed) self-signed certificate.
- With expired or revoked certificate.
- With an expired or revoked CA certificate.

There exist different protocols for enrolment that provide different set of functionalities. Four of them have been briefly described in 6.10.3 through 6.10.6. Because of its simplicity SCEP is broadly use and it is the de-facto-standard in the industry today. Nevertheless, EST can be seen as the successor to SCEP, and provides similar functionality as CMC while keeping the complexity lower through more optional call flows.

An example of enrolling an entity using SCEP is shown in Figure 9.

Notes concerning Figure 9 and Figure 10:

- Prior to enrolment, devices must be configured. Besides their standard configuration, such as their own IP address(es), they must also be given their PKI enrolment data, RA/CA IP address, trust anchor certificates (CA/Root certificate), etc.

- The entity checks the authenticity of the RA/CA based on the root certificate of RA/ CA. This root certificate is feed to the device during the device configuration phase (see 8.1.6). The RA/CA signs the certificates issued the CA private key. Entities check the authenticity of the received certificate by verifying the signature in the certificates using the CA public key (installed in the entities prior to enrolment).
- For SCEP, the activation code is an OTP (One-Time-Password). For EST, the activation code is a username and a password. The username may be the entity's ID that will appear later on its certificate. The password may be any random value (which in essence is similar to an OTP).

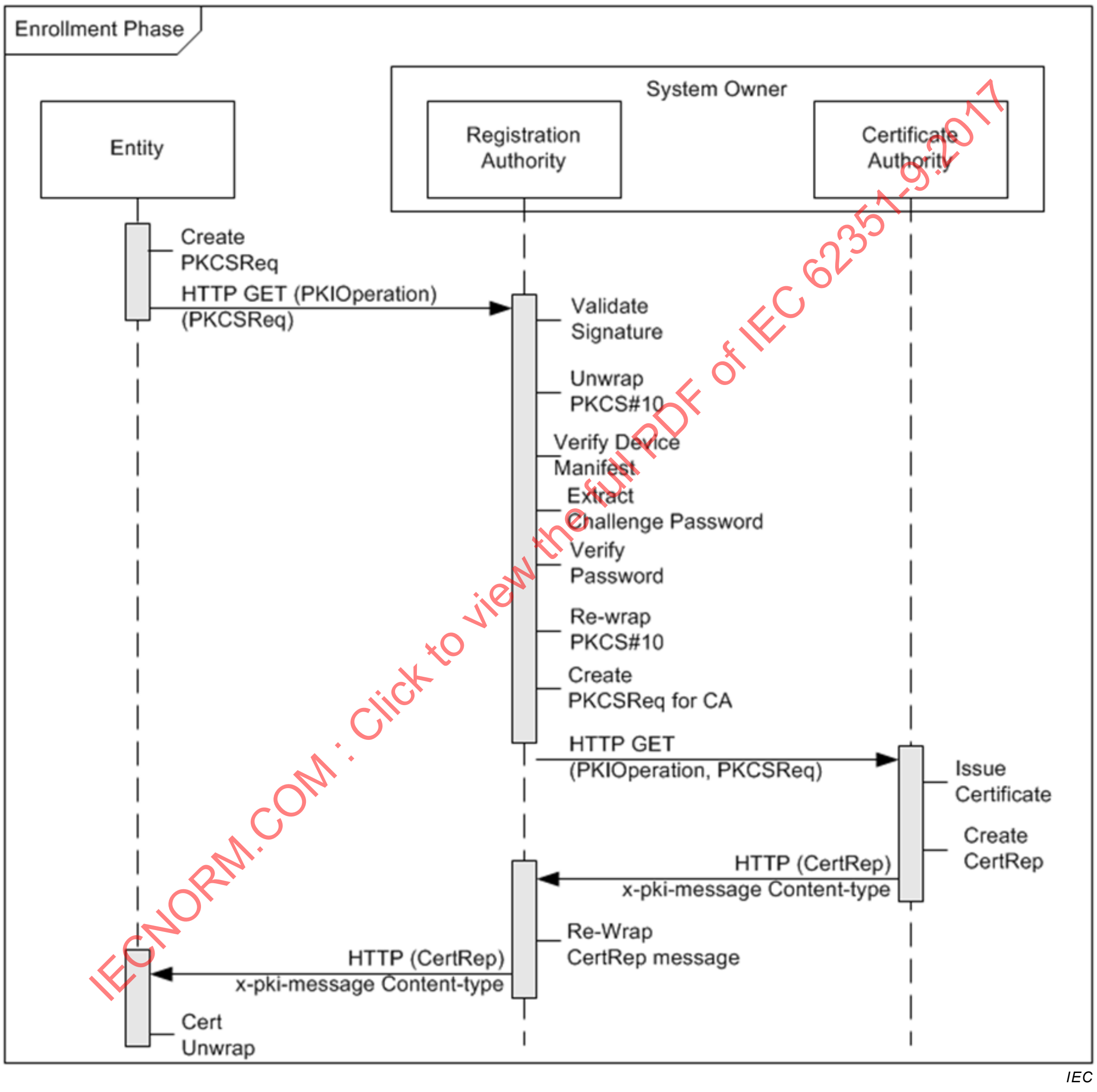


Figure 9 – Example of the SCEP entity enrolment and CSR process

An example of enrolling an entity using EST is shown in Figure 10.

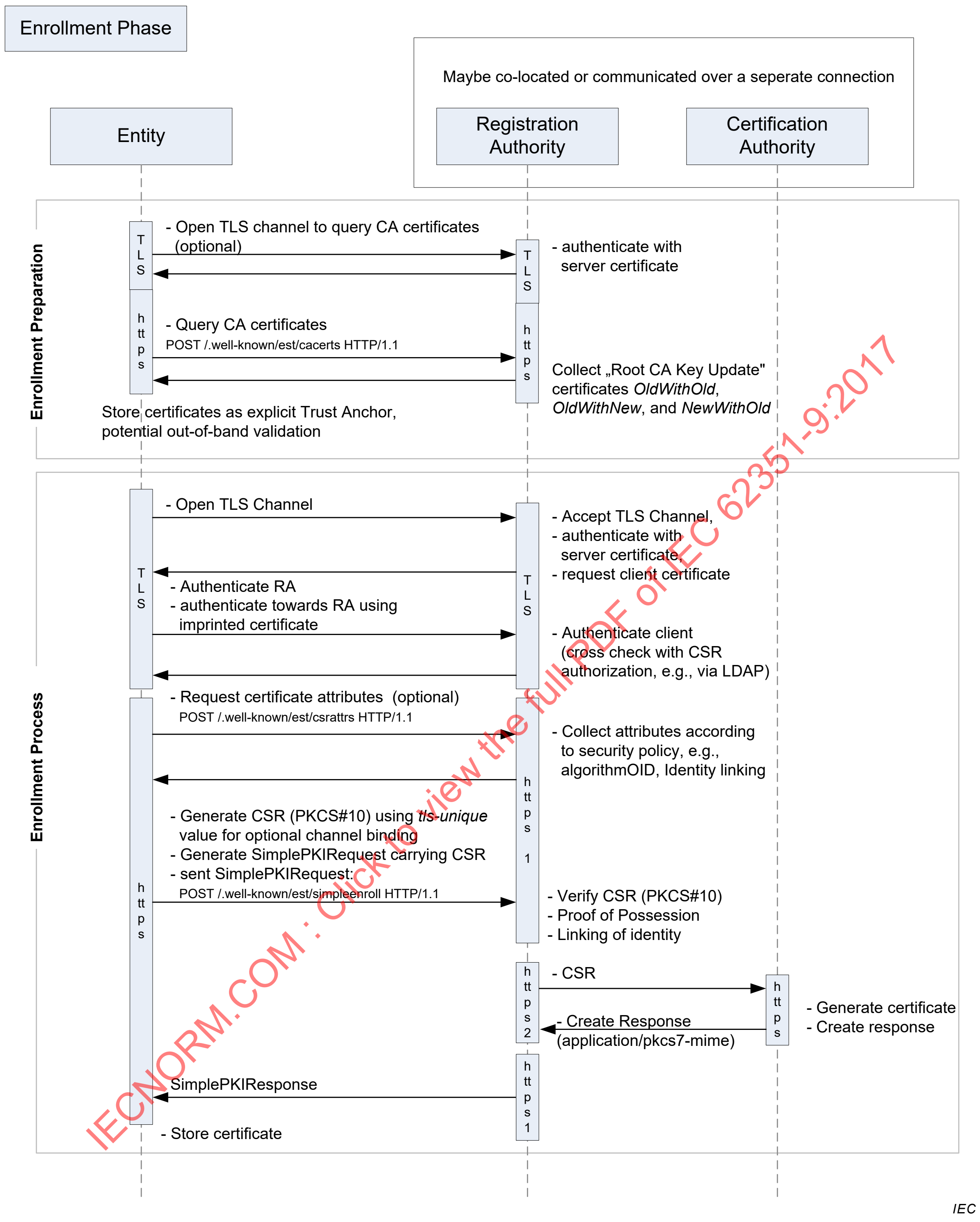

Enrollment Phase
Maybe co-located or communicated over a seperate connection
Entity
Registration Authority
Certification Authority
Enrollment Preparation
TLS
- Open TLS channel to query CA certificates (optional)
- authenticate with server certificate
https
- Query CA certificates
POST /.well-known/est/cacerts HTTP/1.1
Collect „Root CA Key Update" certificates OldWithOld, OldWithNew, and NewWithOld
Store certificates as explicit Trust Anchor, potential out-of-band validation
Enrollment Process
- Open TLS Channel
- Accept TLS Channel, - authenticate with server certificate - request client certificate
- Authenticate RA - authenticate towards RA using imprinted certificate
- Authenticate client (cross check with CSR authorization, e.g., via LDAP)
- Request certificate attributes (optional)
POST /.well-known/est/csrattrs HTTP/1.1
- Collect attributes according to security policy, e.g., algorithmOID, Identity linking
https 1
- Generate CSR (PKCS#10) using tls-unique value for optional channel binding
- Generate SimplePKIRequest carrying CSR
- sent SimplePKIRequest:
POST /.well-known/est/simpleenroll HTTP/1.1
- Verify CSR (PKCS#10)
- Proof of Possession
- Linking of identity
https 2
- CSR
- Create Response (application/pkcs7-mime)
- Generate certificate
- Create response
SimplePKIResponse
- Store certificate
IEC

Figure 10 – Example of the EST entity enrolment and CSR process

6.13.4 Certificate signing request (CSR) process

The enrolment process involves certificate signing by a CA. This certificate signing requires the entity to issue a certificate signing request (CSR) to the RA/CA. One typical CSR process consists of the following steps, as illustrated in Figure 11.

1) The entity generates a pair of public and private keys.

2) The entity generates a *CertificationRequestInfo,* typically using the PKCS#10 specification format ([20]). This request contains a "subject distinguished name", the public key from the public/private pair it just generated (see ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016)), and optionally a set of attributes. The *CertificationRequestInfo* is signed with the entity's private key. The *CertificationRequestInfo*, a signature algorithm identifier, and the entity's signature go into a *CertificationRequest* structure. See [20].
3) The entity sends the *CertificationRequest* message to the RA/CA for authorization and certification.
4) The RA verifies the request by verifying the signature on the incoming request.
5) If the request is valid, the RA authenticates the requesting entity and if valid and authorized invokes the CA, which creates a public-key certificate from the distinguished name and public key, the issuer name, and the certification authority's choice of serial number, validity period, optionally extensions and digital signature algorithm providing the information for the signature of the listed data. The CA then sends the certificate via the RA to the entity.
6) The entity performs the signature verification on the received certificate from the CA to ensure that the received certificate is actually issued and signed by the trusted CA. The trusted CA certificates are distributed and installed in the entity during the commissioning/configuration phase. The entity verifies the signature on the received certificate from CA with the help of these trusted CA certificates. If the CA signature is valid, the entity stores the certificate in the implementation-preferable format (e.g., .der, .pem, .cer, .crt, etc). The PEM format is the most commonly used; it may be easily mapped to other formats.

The CSR process requires some human involvement with an administration role to activate, enable or approve the CSR process. The human involvement may be necessary at the same time as the CSR application or may take place in advance, depending on the entity authentication procedure. If an entity is authenticated with a vendor specific credential or with a one-time password, this information can be provided to the RA in advance.

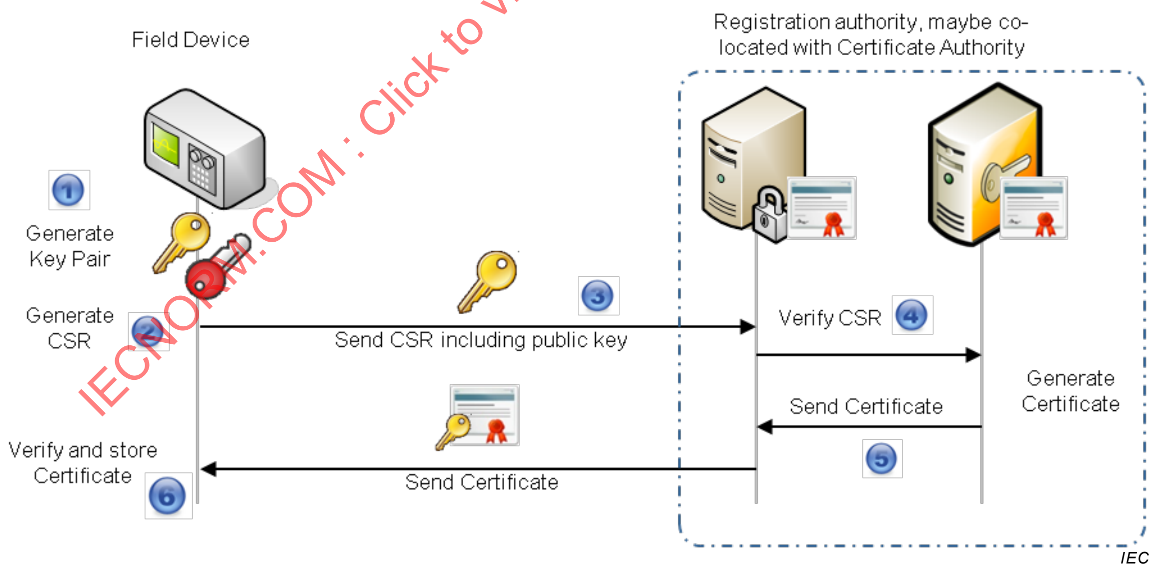

Field Device
Registration authority, maybe co-located with Certificate Authority
1
Generate Key Pair
Generate CSR
2
3
Send CSR including public key
Verify CSR
4
Generate Certificate
Send Certificate
5
Verify and store Certificate
6
Send Certificate
IEC

Figure 11 – CSR processing

6.13.5 Certificate revocation lists (CRLs)

A CRL is a list of the serial numbers of certificates that have been revoked; along with a timestamp indicating when they were revoked, as well as a digital signature from the issuing CA. Revoked certificates should be considered as no longer valid and should not be relied on by any entity.

CRLs should be updated whenever a certificate is revoked. They should be made available to all affected entities in a timely manner. Adequate precision of time synchronization across entities and the system creating CRLs would ensure that the time information in the CRLs is accurate and that the entities have accurate information on revoked certificates. An example of a CRL is shown in Figure 12.

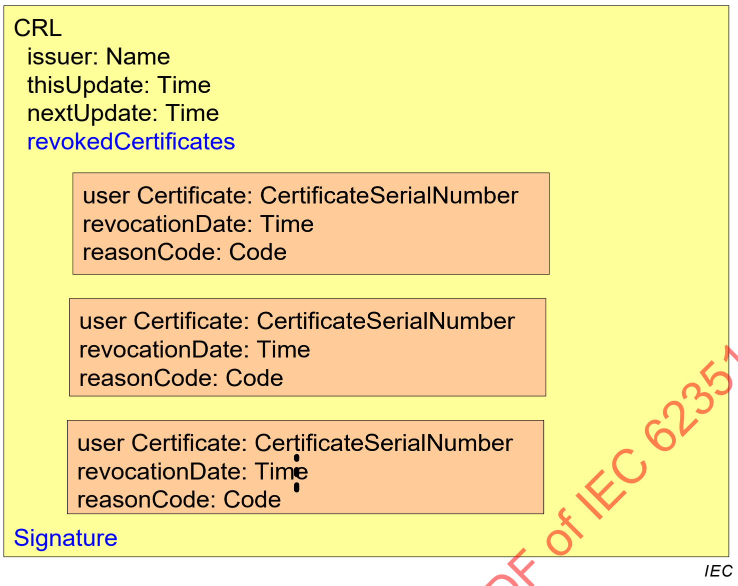


Figure 12 – Certificate revocation list

Since CRLs accumulate revoked certificates over time, they may grow and become very large. Large CRLs could be a problem for embedded systems, since embedded systems might not have enough free memory capacity to store them. CRLs may therefore be partitioned as a means to minimize the list for any particular entity. Alternate methods may be more efficient, as described in 6.13.6.

Certificates should be revoked based on the following reasons and using the reason codes as defined in 9.5.3.1 of ISO/IEC 9594-8:201x | Rec. ITU-T X.5094 (2016).

- The private key is suspected to be compromised.
- The CA private key associated with a CA certificate is suspected to be compromised.
- The entity’s affiliation has changed.
- The public-key certificate has been superseded.
- There has been a cessation of operation of the entity.
- There is a hold on the certificate.
- The privilege has been withdrawn.
- The private key for an attribute authority is suspected to be compromised.

6.13.6 Online certificate status protocol (OCSP)

The online certificate status protocol (OCSP) as defined by RFC 6960, is an alternative to CRL retrieval when checking the revocation status of a public-key certificate.

If a relying party is concerned whether a particular public-key certificate is still valid, it may send an OCSP revocation status request to the OCSP server (or the CA) responsible for the certificate of the entity. This OCSP request contains the protocol version, service request, the entity’s certificate identifier and extensions. In order to avoid replay attacks, a “nonce” (one-time value) is mandatory to distinguish this status request from any previous status requests. The OCSP responder then validates the certificate and returns 'good', 'revoked' or 'unknown', using its own digital signature to authenticate the response.

This OCSP sequence is shown in Figure 13.

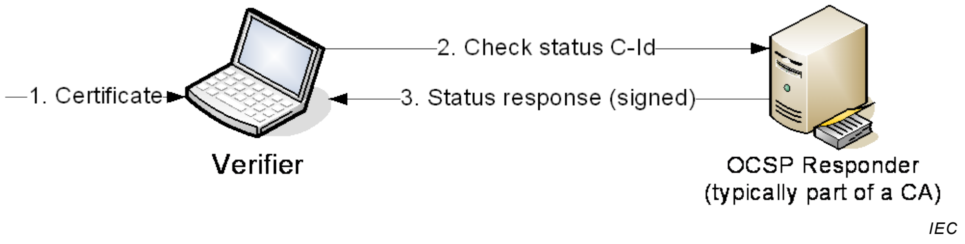


Figure 13 – Overview of the online certificate status protocol (OCSP)

Typically, online connectivity is required between the entity and the OCSP responder for the transmittal of the OCSP request and OCSP response. However, continuous connectivity may be challenging for some configurations of entities in the field. Additionally, the computational effort for processing the OCSP response and the communication delay may not be feasible for some entities. In addition, OCSP responses should have a short validity period since OCSPs do not issue unsolicited status updates, even if a certificate has been revoked shortly after a status check.

Therefore, depending on the system setup and the entity's capabilities, a hybrid combination of CRLs and OCSP may be utilized where one entity that normally does have connectivity acts as a proxy OCSP responder. This proxy entity fetches a CRL within a specific time period such as hourly or within 24 hours. The proxy entity (e.g., station controller) then serves as OCSP responder for other entities that do not normally have connection to the OCSP. This approach is depicted in Figure 14.

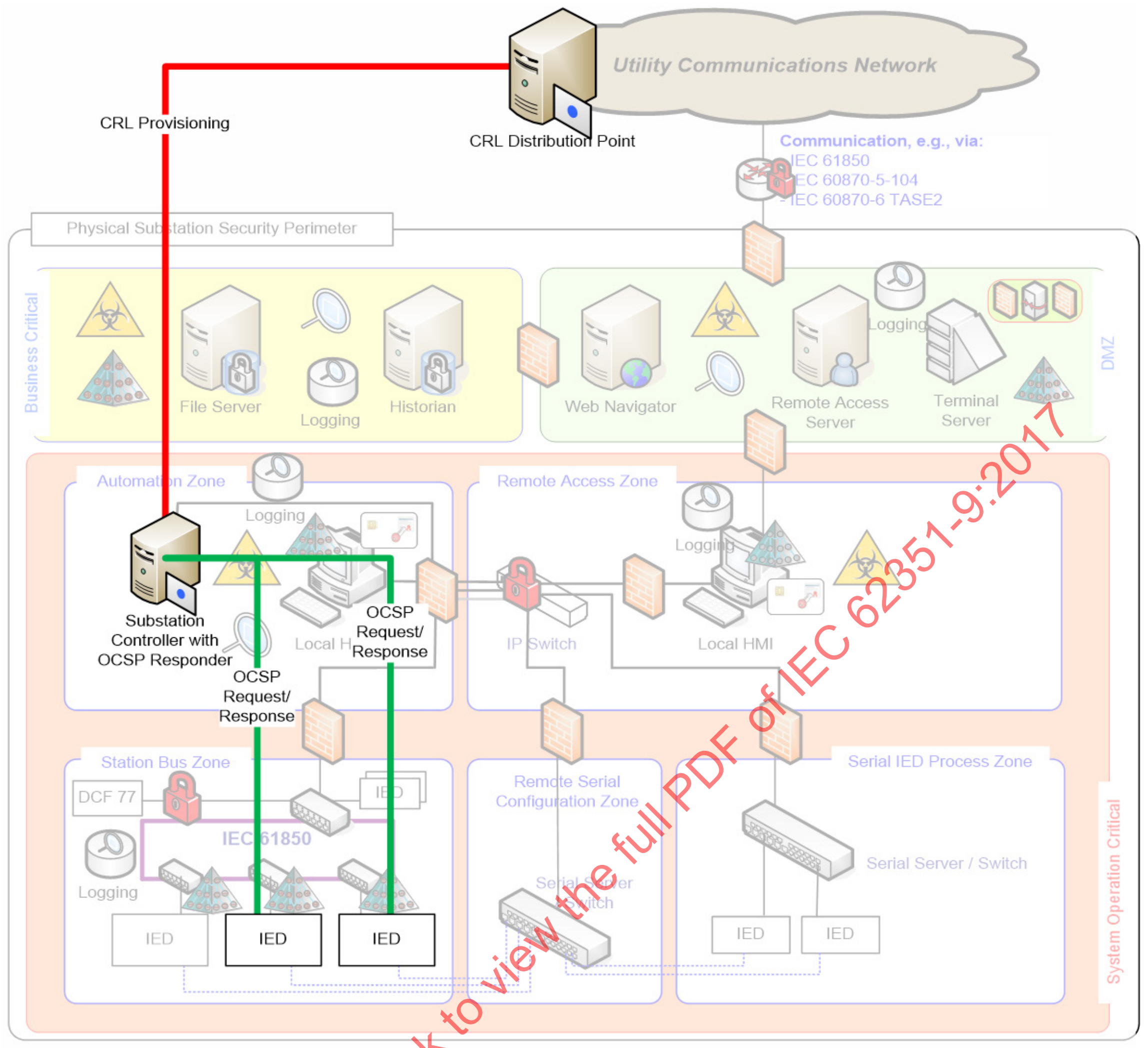


Figure 14 – Diagram using a combination of CRL and OCSP processes

A clear advantage of this approach is that it allows the usage of the OCSP process in local networks that lack permanent connectivity to the central CRL data. Moreover, this reduces the data traffic with the backend, which may be necessary for low bandwidth connections.

Entities should have access to a trusted real-time clock in order to check and assure the accuracy of the OCSP timestamp. If within a configurable timeout (e.g. 15 seconds) no response arrives, entities should trigger an OCSP failure alarm and assume that the certificate has not been revoked (particularly if availability is crucial).

To enable this approach, it is necessary for the proxy OCSP entity (e.g. station controller) to possess a certificate and corresponding private key to act as an OCSP responder. The OCSP responder would thus be configured as trusted responder node with a CA signed certificate.

OCSP requests may also be chained between peer OCSP responders in order to find and query the appropriate CA for the entity certificate being checked, with responders validating each other's responses against the root CA using their own OCSP requests.

OCSP requests may be performed proactively or reactively as depicted in Figure 15. In the first case, the certificate entity requests its status from the OCSP responder in advance and sends the OCSP response together with its certificate to a ”master” node that is validating the entity. This is called “OCSP stapling” and places the burden of OCSP communication on the entity. In the second case, the “master” node performs the OCSP request to check the revocation state of the certificate presented by the entity. The second option is more commonly used, but the first option is supported for protocols like TLS using an extension defined in RFC 6066, allowing the TLS server to transmit an OCSP response as part of the ServerHello message.

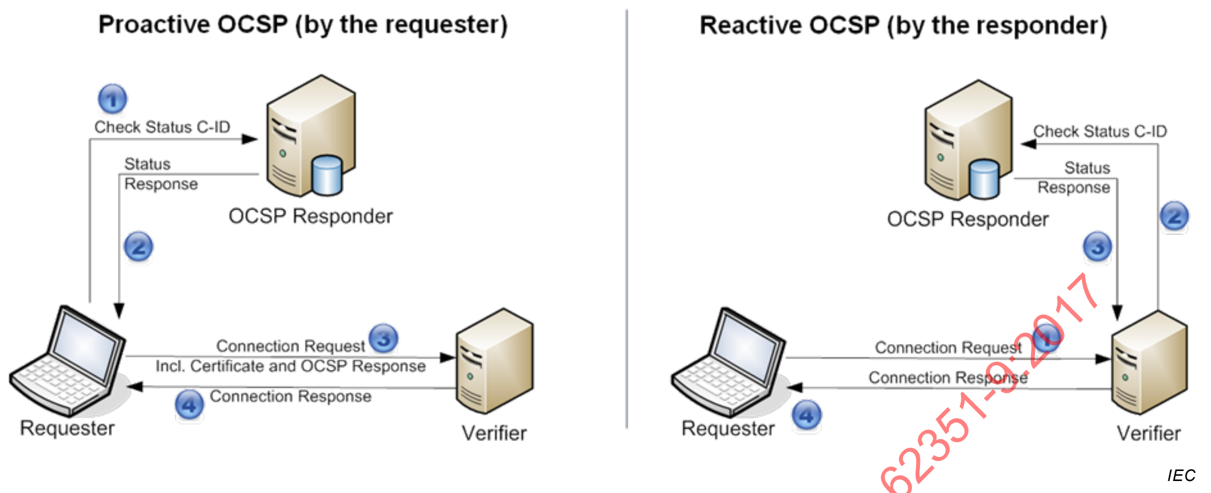


Figure 15 – Call Flows for the Online Certificate Status Protocol (OCSP)

If certificate revocation or certificate validity check fails, it is expected in most cases that the entity will continue operation after sending an OCSP alarm to warn operators about possible unauthenticated communication. This prevents denial-of-service attacks caused by making OCSP responders unavailable.

6.13.7 Server-based certificate validation protocol (SCVP)

The validity check of a certificate may become complex, especially if the certification path is rather long. For this case, the Server-Based Certificate Validation Protocol (SCVP) allows an entity to delegate the certification path construction and certification path validation to a "master" node. This protocol is defined in RFC 5055 ([34]). SCVP may be used in conjunction with CRL as well as with OCSP (as shown in Figure 16).

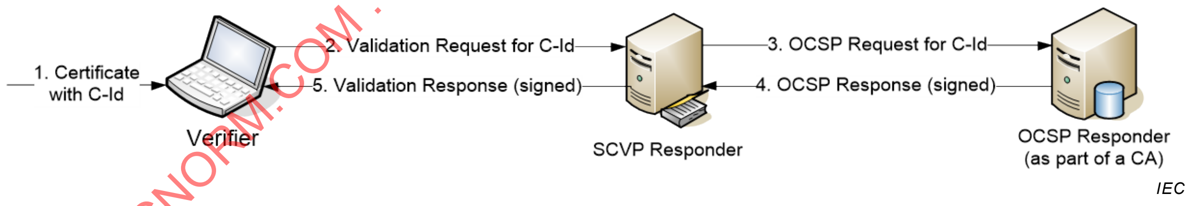

1. Certificate with C-Id
Verifier
2. Validation Request for C-Id
5. Validation Response (signed)
SCVP Responder
3. OCSP Request for C-Id
4. OCSP Response (signed)
OCSP Responder (as part of a CA)
IEC

Figure 16 – Overview Server-Based Certificate Validation Protocol using OCSP Backend

Additionally, in general, it is recommended to keep CA hierarchies as flat and compact as possible to reduce the performance penalties incurred when validating certificate chains.

6.13.8 Short-lived certificates

Short-lived certificates (public-key certificates or attribute certificates) may not need to be revoked, as the possibility for compromise is quite low. Short-lived public-key certificates are probably not relevant, as they may imply additional overhead for issuing and distributing new public-key certificates.

Another approach for short-lived certificates are attribute certificates, which are actually a temporal extension of a public-key certificate as described in 6.5.3. Short-lived attribute certificates may be useful in some applications when, for example, there is a requirement to extend an entity's capabilities in an emergency. Such an attribute certificate should include the no revocation information extension defined in ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

6.13.9 Certificate renewal

When certificates are ready to expire, entities need to apply for certificate renewal using a similar but simpler process than enrolment. Simpler in that the certificate update can already utilize the available certificate information on the respective clients.

An example of SCEP certificate renewal is shown in Figure 17.

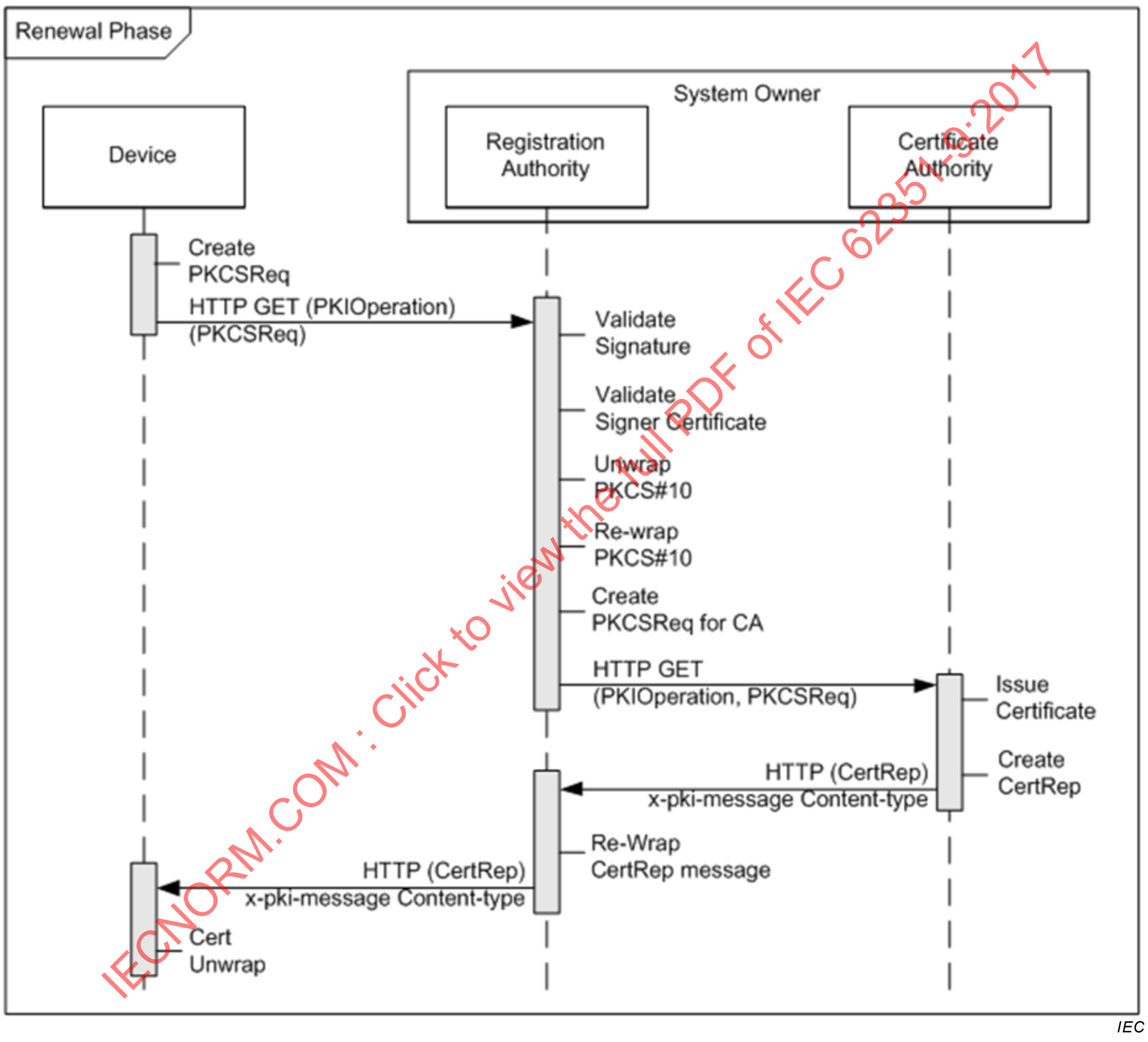


Figure 17 – SCEP certificate renewal

An example of certificate renewal or rekeying using EST is shown in Figure 18.

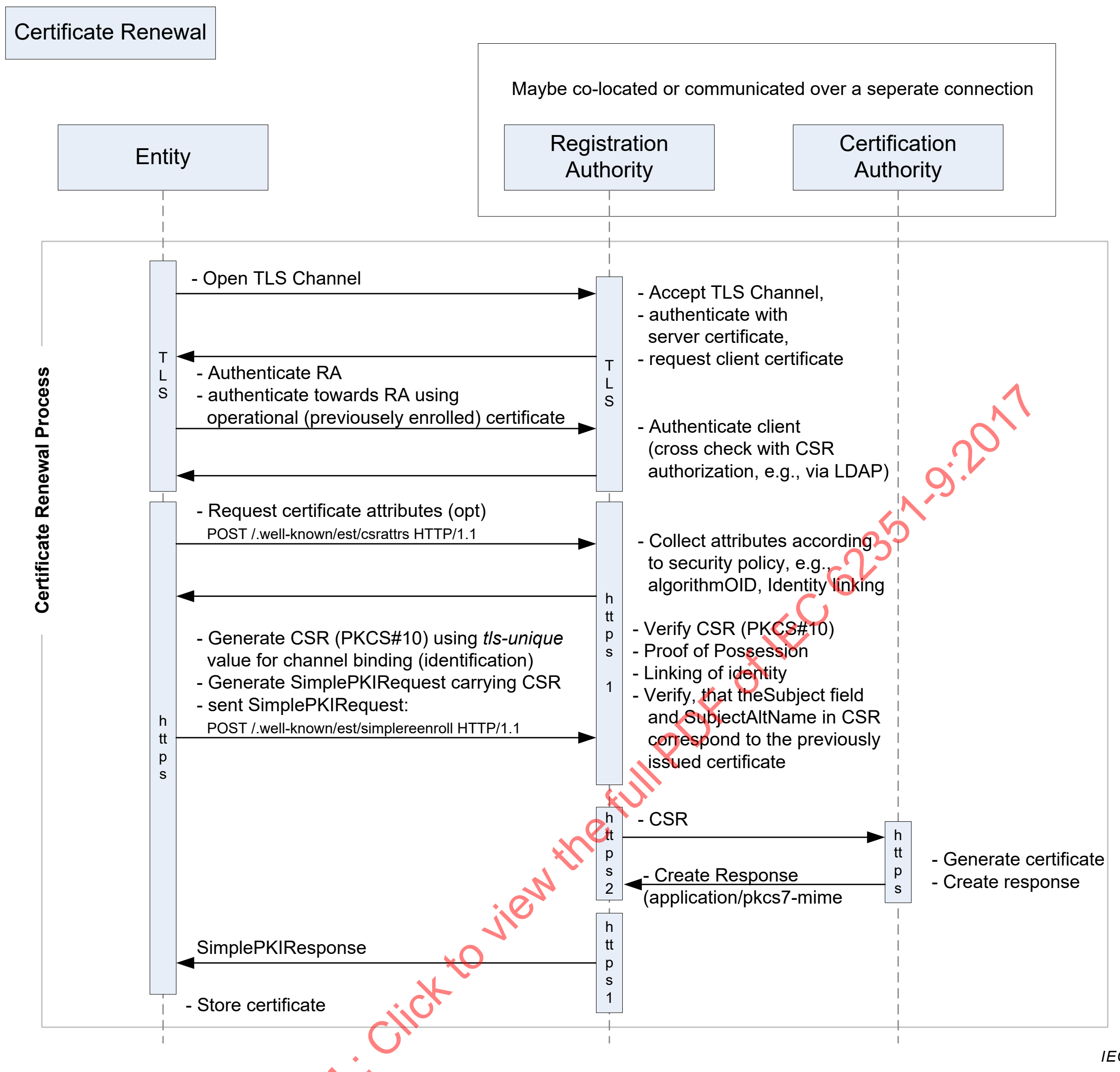

Certificate Renewal
Maybe co-located or communicated over a seperate connection
Entity
Registration Authority
Certification Authority
Certificate Renewal Process
TLS
- Open TLS Channel
- Accept TLS Channel,
- authenticate with server certificate,
- request client certificate
- Authenticate RA
- authenticate towards RA using operational (previousely enrolled) certificate
- Authenticate client (cross check with CSR authorization, e.g., via LDAP)
- Request certificate attributes (opt)
POST /.well-known/est/csrattrs HTTP/1.1
- Collect attributes according to security policy, e.g., algorithmOID, Identity linking
https 1
- Generate CSR (PKCS#10) using tls-unique value for channel binding (identification)
- Generate SimplePKIRequest carrying CSR
- sent SimplePKIRequest:
POST /.well-known/est/simplereenroll HTTP/1.1
- Verify CSR (PKCS#10)
- Proof of Possession
- Linking of identity
- Verify, that theSubject field and SubjectAltName in CSR correspond to the previously issued certificate
https
https 2
- CSR
- Create Response (application/pkcs7-mime
- Generate certificate
- Create response
SimplePKIResponse
- Store certificate
IEC

Figure 18 – EST certificate renewal/rekeying

6.14 Alternative process for asymmetric keys generated outside the entity

Asymmetric key pairs may be generated externally if the entity lacks good random number generation capabilities or the necessary computational power. In that case, the generation process may be delegated to a PKI compliant component, such as a PKI tool. Distribution of the keys would need to be done out-of-band using a trusted procedure. The private key may be protected with a transport key as in PEM, PKCS#8 or usually PKCS#12 that may include other objects, such as CA or root certificates. Figure 19 shows one possible option to realize central key and certificate generation. The distribution of the key material may be done by using a PKI tool locally.

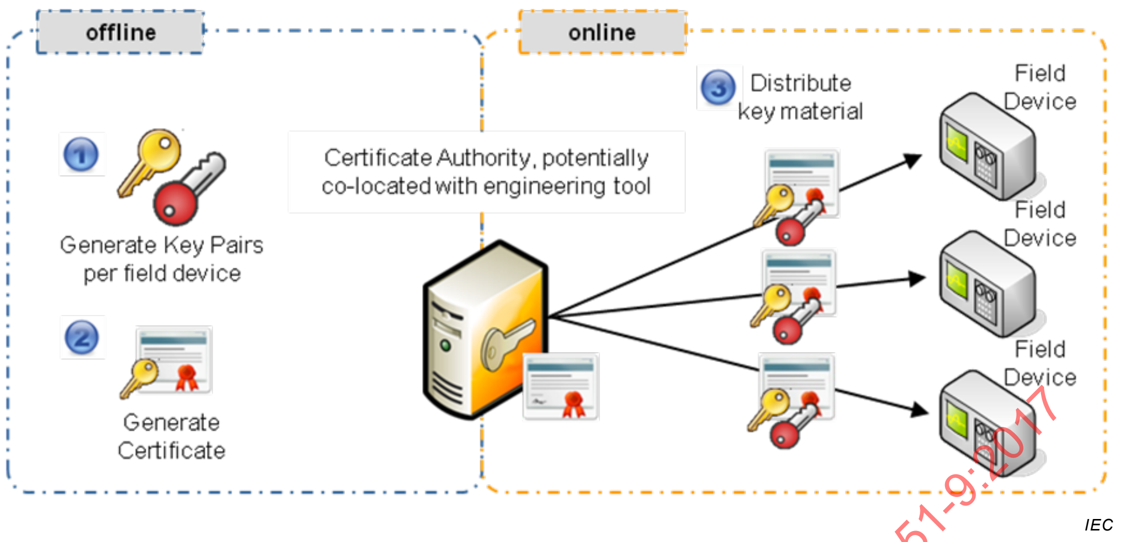

offline
online
1
Generate Key Pairs per field device
2
Generate Certificate
Certificate Authority, potentially co-located with engineering tool
3
Distribute key material
Field Device
Field Device
Field Device
IEC

Figure 19 – Central certificate generation

If a manufacturer-based credential is already available in the field entity and is trusted, this credential may be used to identify the field entity and to secure the key distribution operation. This requires that the credential be already known at the CA.

6.15 Key distribution for symmetric keys with different time frames

Key distribution for symmetric keys may be different for short term and session keys from for long-term keys.

Short term and session keys are typically distributed securely and periodically as part of a dedicated protocol. Long-term keys, such as pre-shared keys, may be administered through system administration, which may be done locally (manually) or remotely.

Different methods may be used for remote symmetric key distribution. These include:

- The PKI method using asymmetric keys to secure the distribution of the symmetric keys between a central site and a single entity (see Clause 8).
- The Group Domain of Interpretation (GDOI) method to secure the distribution of the symmetric keys between a Key Distribution Centre (KDC) and groups of entities.
- As specified in IEC TS 62351-5.
- Using engineering and configuration tools to set entities keys to be used among them.

7 General key management requirements

7.1 Asymmetric and symmetric key management requirements

Key management shall apply to asymmetric key procedures, to symmetric key procedures, and to a combination of both asymmetric and symmetric key procedures. If asymmetric key procedures are used, then Clause 8 shall apply. If symmetric key procedures are used, then Clause 9 shall apply.

7.2 Required cryptographic materials

Standards that reference this document shall specify the required cryptographic materials that shall be present to establish secure communications, as per the Protocol Implementation Conformance Statement PICS form in Annex A.

7.3 Public-Key certificates requirements

The version component shall specify version 3.

Annex D provides a certificate profile stating guidelines on recommended and optional fields to be supported in the utilized certificates.

7.4 Cryptographic key protection

Private keys and long-term symmetric keys (including pre-shared keys) that prove the identity (and Proof of Origin) of each entity shall be protected against unauthorized modification, retrieval and cloning.

During transport, the private key shall be protected against eavesdropping and tampering by being encrypted by a transport key such as is defined in PEM, PKCS#8 and PKCS#12. All entities shall support PKCS#12.

Any attempt to compromise a key shall be detectable if technically possible. Such an event shall be logged and it shall trigger an alarm.

NOTE Some guidance can be found in NIST 800-57, Part 1:2012, 6.2.2 and in FIPS 140-2.

7.5 Use of existing security key management infrastructure

The use of an existing security key management infrastructure (PKI) shall be allowed, so long as the issuing CAs provide complete chain of trust (intermediate CAs) certificates all the way to the root.

7.6 Use of object identifiers

The principle for allocation of object identifiers (OIDs) is specified in ISO/IEC 9834-1:2012 | Rec. ITU-T X.660 (2011). The OID allocation for this part of IEC 62351 is defined as:

id-IEC62351-9 OBJECT IDENTIFIER::= { 1 0 62351 9 }

The following branch is allocated for defining OIDs for AVL extensions:

avl62351Extion OBJECT IDENTIFIER::= { id-IEC62351-9 1 }

The following branch is allocated for defining OIDs for AVL entry extensions:

avl62351EntryExt OBJECT IDENTIFIER::= { id-IEC62351-9 2 }

The following branch is allocated for defining OIDs for protocol identifiers:

id-62351prot OBJECT IDENTIFIER::= { id-IEC62351-9 3 }

8 Asymmetric key management

8.1 Certificate generation and installation

8.1.1 Private and public key generation and installation

Entities performing asymmetric cryptographic functions shall possess at least one pair of asymmetric keys. Either all such entities shall generate their own asymmetric cryptographic key pairs or shall store asymmetric cryptographic key pairs that have been externally generated by a trusted source and have been securely distributed and installed in a protected location.

Entities either shall generate or be provided with new key pairs, via PKCS #12, under the following conditions:

- No key pair is present at start-up time (if not present or the associated public-key certificate has expired).
- Change in controllership of the entity (change in ownership, control authority, and/or reconfiguration of the entity).
- By command from an authorized entity (e.g. request for certificate renewal).
- Entity's private key has been compromised.

8.1.2 Private and public key renewal

New key pairs shall be generated by the entity or shall be provided to the entity within a configurable timeframe before the end of the current certificate validity period. This generation/provision of new key pairs shall lead to a certificate signing request (CSR) in PKI environments.

8.1.3 Random Number Generation

The generation of any random value related to key management shall follow ISO/IEC 19790:2012 [11]. Key pair generators shall be responsible for providing statistically adequate random number generators (RNG) and utilizing them appropriately. Note: Guidance can be found in NIST SP 800-90A [16].

Keys shall be generated externally for entities, which are not capable of generating keys internally. CAs or other authorized systems may provide this external key generation facility. The keys shall then be installed securely in those entities. See Annex B.

8.1.4 Certificate policy

It is strongly recommended to create a certificate policy (see RFC 3647 ([26]) for guidance).

NOTE That Annex D already describes a certificate profile, which is part of the certificate policy.

8.1.5 Entity registration for identity establishment

All entities to be commissioned in an organization shall be registered in at least one registration authority (RA), which may be co-located with the certification authority (CA) that is approved by the organization. This RA shall be able to verify the entities identity on a certificate signing request (CSR). Registration may be done manually (e.g. for a small number of entities), or automatically by running scripts that are generated from engineering data. Registration may take place out-of-band, off-site or on-site.

Registration data shall include at least one of the following elements:

- The entity’s Common Name (CN) which identifies the entity and the unique name that will appear in the entity’s certificate.
- A one-time unique activation code (or OTP), which allows the entity to authenticate itself with the RA, for example, when performing a signature request (CSR). The registration data shall be installed and configured individually into each entity to ensure that the RA can authenticate the entity when it performs a CSR. Note: how the OTPs are created and distributed to the enrolling entities is out of the scope of this standard.
- A certificate serial number and issuer of a manufacturer-build in public-key certificate, which allows the entity to already authenticate using this public-key certificate.
- A fingerprint of the public-key certificate used to authenticate the enrolling entity.

8.1.6 Entity configuration

In addition to the basic certificate parameters as defined in ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016), entity configuration data shall include the following:

- CA certificate(s) of the organization(s) which the entity shall trust and communicate with (see 6.5).
- CA IP address of the organization or a domain name (such as IEC62351.LocalCA) that shall be allowed to perform PKI operations, such as CSR processing.
- CA CSR timeout parameters (such as polling frequency, number of retries, etc.).
- Part of the entity's certificate is the DN (Distinguished Name) and the device's Common Name (CN), which identifies the entity uniquely. This entity name appears in the entity's certificate. It shall be a UTF-8 string with a maximum of 64 bytes.
- Entity's `dnsName`. An entity may list more than one `dnsName`. IP addresses may be used in environments without DNS services.

Either an activation code (a onetime unique password) or a manufacturer built-in certificate shall be supported, which allows the entity to authenticate itself with the CA. Which authentication method is used during enrolment will depend on the system (deployed PKI) capabilities.

8.1.7 Entity enrolment

Once entities have been configured with the required registration data, and are in possession of their own asymmetric key pair, they shall perform a CSR procedure before becoming operational, based on the organization's certificate security policies. An online connection to the organization's RA/CA is required for certificate enrolment, unless out of band enrolment is performed.

Only registered entities shall be allowed to be enrolled at the RA/CA (see 6.13.3).

Entities shall generate the CSR using the PKCS#10 ([20]) format and shall send the CSR to the RA specified during configuration. The RA shall check the validity of the request by verifying the following:

- Proof of possession of the corresponding private key by verifying the CSR signature.
- Proof of identity (by utilizing either the activation code (OTP) or an already available certificate and corresponding private key in conjunction with the registration data on the RA).

If the request is valid, the RA shall send a request to the respective CA. The CA shall generate a public-key certificate and shall send it to the RA, which shall send it to the requesting entity.

If the request is invalid, the RA shall not send any request to the CA.

NOTE 1 The RA and CA may be deployed together as one entity. The process described above still applies.

For automated enrolment, the infrastructure (RA/CA) shall support the following enrolment protocols:

- Simple Certificate Enrolment Protocol (SCEP) for backward compatibility focussing on RSA based public-key certificates.
- Enrolment over Secure Transport, (EST, RFC 7030) for support of RSA or ECC based public-key certificates.

The RA/CA infrastructure is required to support both enrolment protocols, so that a client entity may choose the most appropriate enrolment protocol for the target use case. The decision regarding the utilized enrolment protocol may depend on the required cryptographic algorithms used to issue the certificates. If the environment requires the support of ECDSA, the enrolment protocol shall be EST, as SCEP does not support ECDSA based infrastructures.

The mandatory required functionalities of EST are those described in RFC 7030 and are enhanced with the mandatory support of attribute retrieval. Entities supporting EST shall support:

- the distribution of CA certificates. This functionality allows for the distribution of CA certificates in the initial case to build a trust anchor database, but also the update of the CA certificates;
- the *SimplePKIRequest / Response* (simple enrol) exchange and may support the *FullPKIRequest / Response* exchange;
- the re-enrolment (renewal or re-keying of certificates);
- the *getCACert* message to be able to query a CA certificate based on an implicit trust anchor;
- the *csrattrs* (certificate attribute) message to be able to query CSR certificate attributes to be used in the CSR to allow for explicit requirements from a RA/CA side;
- CSR attribute request/response handling is required to support the selection of signing algorithms and associated parameters (key length, curve selection for elliptic curve based algorithms). The following signing algorithms are permitted:
  - RSA
  - ECDSA
  - ECGDSA

  Associated parameters are key length and selected elliptic curves are explained below.

  NOTE 2 Specific certificate extensions might be specified in the future.

RSA 2048 bit key length shall be supported at a minimum and is the preferred key length to be used. For ECDSA / ECGDSA a key length of 256 bits is recommended. Additionally, for the elliptic curve based algorithms the curves secp256r1 and brainpoolP256r1 are recommended.

NOTE 3 Recommendations regarding required key length for signature algorithms are reviewed constantly and can be found in NIST SP800-57 or BSI TR01202-1.

Optional support for RSA 1024 bit key length is intended for backward compatibility and will be deprecated in the next edition of this standard.

Certificates shall be transported as defined in the enrolment protocols. Certificates may be stored in PEM or PKCS#12 format.

8.1.8 Trust anchor information update

If automated updating of trust anchor information is supported, updating of trust anchor certificates shall be performed using EST (see 8.1.6) or optionally may be performed using the Trust Anchor Management Protocol (TAMP), RFC 5934 [40].

If TAMP is used, the following TAMP messages shall be supported:

- TAMP Status Query
  The TAMP Status Query message is used to request information about the trust anchors that are currently installed in a trust anchor store, and for the list of communities to which the store belongs.
- TAMP Status Query Response
  The TAMP Status Response message is a reply by a trust anchor store to a valid TAMP Status Query message. The TAMP Status Response message provides information about

the trust anchors that are currently installed in the trust anchor store and the list of communities to which the trust anchor store belongs, if any.

- Trust Anchor Update
The Trust Anchor Update message is used to add, remove, and for change management and identity trust anchors. The Trust Anchor Update message cannot be used to update the apex trust anchor.
- Trust Anchor Update Confirm
The Trust Anchor Update Confirm message is a reply by a trust anchor store to a valid Trust Anchor Update message. The Trust Anchor Update Confirm message provides success and failure information for each of the requested updates.
- Apex Trust Anchor Update
The Apex Trust Anchor Update message replaces the operational public key and, optionally, the contingency public key associated with the apex trust anchor. Each trust anchor store has exactly one apex trust anchor. No constraints are associated with the apex trust anchor. The public key identifier of the operational public key is used to identify the apex trust anchor in subsequent TAMP messages. The digital signature on the Apex Trust Anchor Update message is validated with either the current operational public key or the current contingency public key per the RFC.
- Apex Trust Anchor Update Confirm
The Apex Trust Anchor Update Confirm message is a reply by a trust anchor store to a valid Apex Trust Anchor Update message. The Apex Trust Anchor Update Confirm message provides success or failure information for the apex trust anchor update.
- TAMP Error
The TAMP Error message is a reply by a trust anchor store to any invalid TAMP message. The TAMP Error message provides an indication of the reason for the error.

8.2 Public-key certificate revocation

The public key infrastructure (PKI) servers and certificate distribution points (CDPs) shall support at a minimum the following secure methods for revocation:

- Certificate Revocation Lists (CRL pulled via SCEP or LDAP or HTTP or other).
- Online Certificate Status Protocol (OCSP).

The PKI CAs shall propagate revocation information, at a minimum, every 24 hours (this may vary depending on, PKI policies, criticality of particular devices and number of peer connections).

System behaviour in case of unavailability of CRLs updates or of OCSP responders should be part of the organization's certificate security policy.

8.3 Certificate validity

8.3.1 Validity of certificates

Entity certificates, including public-key certificates, attribute certificates, and CA certificates, shall be validated before entities are trusted. This validation shall include, at a minimum, the following checks:

- The validity period of the entity's digital certificate. Access to a clock that is synchronized to Time Atomic International (TAI) or UTC is required in order to assure accurate assessment of the certificate's expiration date. Accurate time and clock synchronization in a network is typically provided by the NTP or PTP protocols, which are defined respectively in RFC 5905 [39] or IEEE 1588 [6]; Note that both protocols are currently revised and will include advanced security options.
- The digital signature of entity's public-key certificate from the issuing CA as well as the certificate path down from the trust anchor.
- The revocation status of the entity's public-key certificate.

Additional checks should follow the certificate policy of the entity’s organization.

It is recommended to validate self-signed certificates using certificate white listing.

8.3.2 Certificate revocation

An entity’s certificate shall be revoked at a minimum under the following conditions, using the reason codes specified in 9.5.3.1 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

- Entity’s private key has been compromised.
- CA has been compromised.
- Entity’s affiliation has changed.
- Entity’s end of life.

Additional reasons for the revocation of a certificate may be provided by the revocation policy of the organization. It is not required to revoke certificates that have been renewed or that have expired.

Revocation is a requirement of the PKI environment and not of a specific entity.

8.3.3 Certificate revocation status checking

Entities shall implement one or more of the following technologies to ascertain the validity status of digital certificates:

- Certificate Revocation Lists (CRL).
- Online Certificate Status Protocol (OCSP).

The specific method used is dependent upon the requirements of the specific protocol being implemented, i.e. IEC 62351 Part 3 requires end entities support CRLs.

In the context of TLS, OCSP response may be included as part of the TLS handshake from the server side (a.k.a. OCSP Stapling, cf. RFC 6066).

8.3.4 Handling of authorization and validation lists (AVLs)

8.3.4.1 General

Support of AVLs and associated protocols is optional, but if used shall meet the requirements in ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016) as follows:

a) Clause 11 specifies AVLs.

b) Clause 20 specifies a wrapper protocol similar to the cryptographic message syntax (CMS) as defined in RFC 5652 [37], but is somewhat simpler to provide a lightweight protocol. This wrapper protocol is used for transport of information for the management of AVLs.

c) Clause 21 specifies protocols for supporting the management of AVLs. It specifies two protocol using the services of the wrapper protocol:

   1) The authorization and validation management protocol (AVMP) is used for communication between an authorizer and the AVL entities its support.

   2) The CA subscription protocol (CASP) used by an authorizer to subscribe to public-key certificate status information from relevant CAs. This protocol is only used for constrained environments (see 6.7.3). This protocol requires that CAs are updated to support the CASP protocol and associated capabilities.

8.3.4.2 Syntax for authorization and validation list (AVL) for public-key certificates

The `CertAVL` is the data type specifying the syntax of an AVL and is defined as follows:

```
CertAVL::= SIGNED {TBSCertAVL}

TBSCertAVL::= SEQUENCE {
  version                [0]  IMPLICIT Version DEFAULT v1,
  serialNumber                AvlSerialNumber OPTIONAL,
  signature                   AlgorithmIdentifier {{SupportedAlgorithms}},
  issuer                      Name,
  constrained                 BOOLEAN,
  entries                     SEQUENCE (SIZE (1..MAX)) OF SEQUENCE {
    idType                      CHOICE {
      certIdentifier        [0]   PKCertIdentifier,
      entityGroup                 DistinguishedName, – only for constrained = FALSE
      ... },
    scope                  [0]  IMPLICIT ScopeRestrictions OPTIONAL,
    entryExtensions        [1]  IMPLICIT Extensions OPTIONAL,
    ... },
  ...,
  ...,
  avlExtensions               Extensions OPTIONAL }

AvlSerialNumber::= INTEGER (0..MAX)

PKCertIdentifier::= CHOICE {
  issuerSerialNumber          IssuerSerialNumber,
  fingerprintPKC         [0]  IMPLICIT FingerPrint {Certificate},
  fingerprintPK          [1]  IMPLICIT FingerPrint {PublicKey},
  ... }

IssuerSerialNumber::= SEQUENCE {
  issuer        Name,
  serialNumber  CertificateSerialNumber,
  ... }

ScopeRestrictions::= SEQUENCE OF ScopeRestriction

SCOPE-RESTRICTION::= TYPE-IDENTIFIER

ScopeRestriction::= SEQUENCE {
  id            SCOPE-RESTRICTION.&id,
  restriction   SCOPE-RESTRICTION.&Type,
  ... }
```

The **TBSCertAVL** data type specifies the actual syntax of an AVL. **SIGNED** is a so-called parameterized data type defined in 6.2.1 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016) as

```
SIGNED{ToBeSigned}::= SEQUENCE {
  toBeSigned    ToBeSigned,
  COMPONENTS OF SIGNATURE,
  ... }
```

where the **SIGNATURE** is a sequence of the signature algorithm used and the digital signature.

The different components of the **TBSCertAVL** data type are described in Clause 11 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016). Only the specific AVL aspects relevant to power systems are described below:

a) The **serialNumber** component is provided to allow more advanced functionality by allowing an authorizer to place more than one AVL in an AVL entity. It is also possible that several authorizers place one or more AVLs in the same AVL entity.

b) The **constrained** component shall take the value **FALSE** to indicate that AVL entity is a non-constrained AVL entity. This value should be set until such time that CAs support public-key certificate status subscription.

c) The **entries** component shall hold an element for each public-key certificate and/or entity group represented by the AVL. Each element shall be specified as follows:

1) The **idType** component shall take one of two alternatives:
   i) If the **certIdentifier** alternative is taken, it shall identify the particular public-key certificate represented by this entry. It can be done by specifying the CA issuer name together with the serial number of the public-key certificate or it may be identified by a fingerprint of the public-key certificate or just the public key. If the public-key certificate is a self-signed certificate, only a fingerprint may be used for identifying the public-key certificate.
   ii) If **entityGroup** alternative is taken, it shall hold that part of a distinguished name that is common for all the entities in the group. This alternative is only relevant for a non-constrained environment.
2) The **entryExtensions** component, when present, shall hold one or more extensions specific for the entry in question. The extensions specified in 8.3.4.4 to 8.3.4.6 may be included here. If a particular extension type is used as an entry extension, it shall not be included in the **avlExtensions** component.

d) The **avlExtensions** component, when present, shall hold one or more extensions applicable for entries in the AVL. The extensions specified in 8.3.4.4 to 8.3.4.5 may be included here. If a particular extension type is included here, it shall not be present in the **entryExtension** component of any entry.

8.3.4.3 General on AVL extensions for power systems

As seen from the **TBSCertAVL** data type, extensions may be added to both the individual entries and the AVL as a whole. Such extensions are specified in 8.3.4.4 to 8.3.4.6.

8.3.4.4 Scope restriction

In certain scenarios it may be desired also to include the scope into a public-key certificate, this is an optional extension. The scope is intended to limit the applicability of public-key certificates. The actual scope constraints are defined in a separate type. This allows using the constraint also separately.

The **scopeConstraints** extension syntax is defined as:

```
scopeConstraints EXTENSION::= {
  SYNTAX        ScopeConstraints
  IDENTIFIED BY { avl62351Extion 1 } }

ScopeConstraints =:: SEQUENCE Of (SIZE (1..MAX)) OF ScopeConstraint

ScopeConstraints::= SEQUENCE { – contains the scope information
  aor        UTF8String (SIZE(1..64)),
  --Def. of "Area of Responsibility" of CA cert
  revision   INTEGER (0..255) OPTIONAL
  – optional revision if aor changes
}
```

The area of responsibility (**aor**) restricts the applicability of a public-key certificate to a certain geographical or organizational area. This standard defines the field and the format for the **aor** as follows:

Field name	Coding, max length (byte)	Example
Area of responsibility	UTF8, 64	DE.BAVARIA

The **aor** is an identifier and defines a hierarchical name space or a reference to the namespace. Note that these identifiers are typically alphanumeric. The **aor** would be provided per policy, e.g., from the responsible operator.

NOTE The notion of **aor** (or scope) to describe a geographical or organizational restriction has already been introduced in IEC 62351-8 and is being reused here.

8.3.4.5 Protocol restriction extension

This scope restriction is defined in 11.4 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016). It is used to enumerate associated protocols to be allowed to be used when communication with the entity (or entities in case of **entityGroup**). If this scope restriction is omitted, there are no restrictions with respect to the types of protocols to be employed. The following is the object identifier allocation for important protocols for power system.

Note, that “T” denotes the application on transport level, while “A” denotes to the application level. Conforming authorizers shall flag this extension as non-critical.

```
id-P-IEC61850-T   OBJECT_IDENTIFIER::= { id-IEC62351prot 1 }
id-P-IEC61850-A   OBJECT_IDENTIFIER::= { id-IEC62351prot 2 }
id-P-IEC60870-5-T OBJECT_IDENTIFIER::= { id-IEC62351prot 3 }
id-P-IEC60870-5-A OBJECT_IDENTIFIER::= { id-IEC62351prot 4 }
id-P-IEC62325-T   OBJECT_IDENTIFIER::= { id-IEC62351prot 5 }
id-P-IEC62325-A   OBJECT_IDENTIFIER::= { id-IEC62351prot 6 }
id-P-IEEE1518-T   OBJECT_IDENTIFIER::= { id-IEC62351prot 7 }
id-P-IEEE1518-A   OBJECT_IDENTIFIER::= { id-IEC62351prot 8 }
```

8.3.4.6 Pinning of Certificate and associated Identifier

This AVL **pinningId** entry extension provides an association of a distinct identifier to a public-key certificate. This identifier is most likely the IP address, but may also be another identifier. There are use cases in which a public-key certificate is only to be used on a dedicated IP address. The **pinningId** extension supports the association of a certificate with the IP address (or another identifier), if the certificate itself does not provide IP address information as part of the in **subjectAltName** extension

```
pinningId EXTENSION::= {
  SYNTAX        PinningId
  IDENTIFIED BY { avl62351Extion 2 } }

PinningId::= GeneralNames

GeneralNames::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName::= CHOICE {
  otherName                 [0]  OtherName,
  rfc822Name                [1]  IA5String,
  dNSName                   [2]  IA5String,
  x400Address               [3]  ORAddress,
  directoryName             [4]  Name,
  ediPartyName              [5]  EDIPartyName,
  uniformResourceIdentifier [6]  IA5String,
  iPAddress                 [7]  OCTET STRING,
  registeredID              [8]  OBJECT IDENTIFIER,
  ... }
```

The **GeneralNames** data type is defined and the alternatives of **GeneralName** are described in 9.3.2.1 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016). The **GeneralNames** data type is copied here for easy reference.

The **iPAddress** shall be stored in the octet string in "network byte order", as specified in RFC 791. The least significant bit (LSB) of each octet is the LSB of the corresponding byte in the network address. For IP version 4, as specified in RFC 791, the octet string shall contain exactly four octets For IP version 6, as specified in RFC 2460, the octet string shall contain exactly sixteen octets.

8.3.4.7 Public-key certificate extensions related use of AVLs

8.3.4.7.1 General

Especially if AVLs are used in a PKI context, certain public-key certificates extensions are recommended to be used. Note that if self-signed certificates are used, the inclusion of these extensions may not be enforceable in practice.

8.3.4.7.2 AVL distribution point extension

The AVL Distribution Point extension defines how AVL information can be obtained. It is modelled after the CRL distribution points and may be used optionally.

```
aVLDistributionPoints EXTENSION::= {
  SYNTAX          AVLDistributionPoints
  IDENTIFIED BY   id-aVLDistributionPoints }

AVLDistributionPoints::= SEQUENCE SIZE (1..MAX) OF AVLDistributionPoint

AVLDistributionPoint::= SEQUENCE {
  distributionPoint        [0] AVLDistributionPointName OPTIONAL,
  aVLIssuer                [1] GeneralNames OPTIONAL }

AVLDistributionPointName::= CHOICE {
  fullName                 [0] GeneralNames,
  nameRelativeToAVLIssuer  [1] RelativeDistinguishedName }

id-aVLDistributionPoints OBJECT IDENTIFIER::= { avl62351Extion 1 }
```

The definition of the protocol to be used is outside the scope of this document.

This extension should only to be used if all AVL entities having to validate this public-key certificate are managed by the same authorizer.

In the more general case, the procedure specified in 21.4.2 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016) may be used. In this procedure, the authorization and validation protocol is used by an AVL entity to initiate communication with its authorizer and thereby invite the authorizer to submit the AVL to be used by AVL entity. This procedure requires some initialization information to be provided to the AVL entity, e.g. during the engineering process.

8.3.4.7.3 Authorization and validation extension

The Authorization and validation extension is specified in 9.2.2.8 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016). This extension is a public-key certificate extension. The present of this extension in a public-key certificates signals that this public-key certificate shall only be considered valid if it can be checked against a particular AVL.

This extension shall always be flagged as critical.

This extension should only to be used if all AVL entities having to validate this public-key certificate are managed by the same authorizer (this limitation is not mentioned in. ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016)).

NOTE If this extension is not present, it is a security policy decision whether to accept this public-key certificate.

8.3.4.7.4 Extended Key Usage

The authorizer uses its private key for signing an AVL to be submitted to an AVL entity. The corresponding public-key certificate shall include the extended key usage extension with the value `id-avlSign` (= 2.5.38.2). This extended key usage extension and the `id-avlSign` value is defined in 9.2.2.4 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

8.3.4.8 Issuing of an AVL

Issuing an AVL is the responsibility of the operator (authorizer) of a dedicated power system. The AVL is expected to be compiled based on the engineering data for a specific system deployment. Based on the engineering data the communication relations are known as well as the protocols used between the communicating entities. The AVL can be compiled at the same time, assumed, that entity specific public-key certificates are already available. If entity specific components are not available at engineering time, the CA issuing the certificates may be configured with the engineering information and provide this information to the entities during the enrolment.

Note that an AVL needs to be updated, if the communication peers in the system change. This may be the case if components of the system are removed or exchanged or newly introduced. It is assumed that at least the removal or introduction of components is accompanied by engineering.

Clause 21 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016) specifies procedures for management of AVLs.

8.3.4.9 Endpoint Handling of AVLs

The process of verifying a public-key certificate received during a connection establishment (e.g., during the TLS handshake) and the validation of the public-key certificate itself and the check against a locally available AVL is outlined in IISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

8.4 Certificate expiration and renewal

This document imposes neither a minimum nor maximum public-key certificate life span. A certificate expiration date should be chosen depending on the certificate type and on the local security policies (see 8.1.4).

Public-key certificates may optionally include a private key usage extension, which specifies the period during which the corresponding private key may be used by its owner. This period is normally set to be shorter than the certificate validity period ensuring that certificates remain valid for a minimum period after use by their owner. Details on the use of the private key usage extension may be found in 9.2.2.5 of ISO/IEC 9594-8:2017 | Rec. ITU-T X.509 (2016).

Entities shall generate a new key pair and perform a CSR after their public-key certificates expiration dates get to a certain percentage of the public-key certificate life span, as specified by the organization’s certificate policies. Entities shall renew their public-key certificates before they expire and shall log their public-key certificate renewal actions (as successful or failed events).

Entities shall allow the public-key certificate renewal policy to be configured, such as:

a) Auto-renewal is supported or not.
b) The length of time before expiration that public-key certificate renewal shall take place.

8.5 Secured Time Synchronization

Clock synchronization and accuracy shall be ensured. Time synchronization shall be implemented using the security options of IEEE 1588 or IETF’s NTS, when they become available, or using other security protocols.

9 Symmetric key management

9.1 Group based key management (GDOI)

9.1.1 GDOI requirements

An informative overview of GDOI is provided in 6.11.

The Group Domain of Interpretation (GDOI) method, RFC 6407 ([44]) and the Internet Draft for "GDOI Protocol Support for IEC 62351 Security Services" [52] shall be used for the distribution of group keys to Group Members (GM). The Internet Draft for "GDOI Protocol Support for IEC 62351 Security Services" is hereafter referred to as the IEC 62351 ID.

The GROUPKEY-PULL method is mandatory and the implementation of GROUPKEY-PUSH is optional.

The capability to use the GDOI client certificate credentials exchanged at connection establishment time is mandatory.

The KDC shall maintain the repository of the keys and associated parameters in a secure location.

The KDC shall configure the policy for renewing the session key, which shall be based on key lifetime. A GM may override this policy with a policy based on a number of messages sent with the current key if the trigger occurs before the lifetime of the current key expires

9.1.2 Internet Key Exchange Version 1 (IKEv1)

GDOI RFC 6407 provides an ISAKMP implementation where GDOI is a new ISAKMP Domain of Interpretation (DOI). RFC 6407 defines how IKEv1 (RFC 2409) is used as the phase 1 protocol for GDOI. The KDC shall use IKEv1 for its GDOI Phase 1 protocol. The KDC does not need to support all of the functionality and features of IKEv1, but at a minimum shall support the IKEv1 requirements listed in Table 1.

Table 1 – KDC IKEv1 Requirements

Description	Values
ISAKMP Exchanges Supported	2 – Main Mode (ID Protection) 5 – Informational
GM ID Payload Types Supported	9 – ID_DER_ASN1_DN (Subject ID of the GM's Identity Certificate)
Key Exchange Supported	Diffie-Hellman via Group Description Attribute
Server IP Port	UDP 848 (configurable)
1 – Encryption Algorithm Attribute (Mandatory)	5 – 3DES-CBC 7 – AES-CBC (Key Lengths: 128/256)
2 – Hash Algorithm Attribute (Mandatory)	4 – SHA2-256 5 – SHA2-384 6 – SHA2-512
3 – Authentication Method Attribute (Mandatory)	3 – RSA Signatures
4 – Group Description Attribute (Mandatory)	2 – MODP-1024 5 – MODP-1536 14 – MODP-2048 15 – MODP-3072 16 – MODP-4096
11 – Life Type (optional)	1 – Seconds
12 – Life Duration (optional)	120:86400 Seconds (default – 120)
14 – Key Length	Required if AES-CBC encryption algorithm is used.

9.1.3 Phase 1 IKEv1 main mode exchange type 2

9.1.3.1 General

IKEv1 (RFC 2409) defines two methods to perform a phase 1 exchange: “main mode” and “aggressive mode”. Support for IKEv1 main mode exchange is mandatory and support for IKEv1 aggressive mode exchange is prohibited. IKEv1 main mode exchange is an instantiation of the ISAKMP identify protection exchange. IKEv1 main mode exchange uses ISAKMP Exchange Type 2.

IKEv1 specifies four main mode exchanges depending on the authentication method (IKE authentication methods SA attribute, value 3) provided in the initial SA payload:

1) Authentication with digital signatures
2) Authentication with public key encryption
3) Revised method of authentication with public key encryption
4) Authentication with a pre-shared key

Support for authentication with RSA digital signatures is mandatory. The RSA digital signature is IKEv1 authentication methods attribute value 3. Authentication with DSA digital signatures, public key encryption and pre-shared keys are optional.

The main mode exchange with RSA digital signatures is defined in Figure 20.

```
      Initiator (GM)                          Responder (KDC)
     ---------------                         -----------------
      HDR, SA                        -->
                                     <--      HDR, SA
      HDR, KE, Ni                    -->
                                     <--      HDR, KE, Nr
      HDR*, IDii, [ CERT, ] SIG_I    -->
                                     <--      HDR*, IDir, [ CERT, ] SIG_R

        * Protected by the Phase 1 SA; encryption occurs after HDR
```

IEC

Figure 20 – IKEv1 (RFC 2409) main mode exchange with RSA digital signatures

As depicted in Figure 20, the Group Member is always the initiator of the exchange. The KDC is always the responder of the exchange. The notations (i.e. HDR, SA, KE, etc.) are taken from RFC 2409. Please refer to RFC 2409 section 3[2] for definitions of the notations.

Subclauses 9.1.3.2 to 9.1.3.5 describe each message and notes where the KDC differs from or limits the IKEv1 protocol.

9.1.3.2 Certificate request payload

The GM shall be compliant with ISAKMP 2408 and IKEv1 2409 with respect to how certificate request payloads are used. The GM shall send a certificate request payload as defined in RFC 2408, section 3.10 and shall return the requested certificate in a subsequent message response. If the KDC does not receive a message with a certificate request payload, the KDC shall not send its certificate. The KDC may add a certificate request payload in the second

2 http://tools.ietf.org/html/rfc2409#page-3.

exchange message sent to the GM. If the KDC does not send a certificate request payload, the GM shall not send its certificate.

9.1.3.3 Security association exchange (1)

9.1.3.3.1 General

The first two messages exchanged shall be used to determine the security association for the IKE SA, as shown in Figure 21.

```
     Initiator (GM)                          Responder (KDC)
    ---------------                         -----------------
    (1) HDR, SA                      -->
                                    <--      HDR, SA
    (2) HDR, KE, Ni                  -->
                                    <--      HDR, KE, Nr
    (3) HDR*, IDii, CERT, SIG_I      -->
                                    <--      HDR*, IDir, CERT, SIG_R
```
IEC

Figure 21 – IKEv1 main mode exchange and security association messages

The GM shall send a list of proposed transforms in order of precedence and the KDC shall choose the first compatible one.

The initial message sent by the GM shall contain one SA payload as defined in RFC 2409, section 5 *Exchanges*. An IKEv1 SA payload is defined as one or more transform payloads embedded in a proposal payload that is embedded in an SA payload. Multiple proposal payloads shall not be supported.

9.1.3.3.2 SA payload

The SA payload (SA) shall be the same as RFC 2408, section 3.4. The KDC shall support GDOI RFC 6407section 2.1 which requires that the phase 1 SA payload DOI field shall be set to GDOI (2). The SA payload Situation field is 4 octets and shall be set to 0. Any other values shall result in an error.

9.1.3.3.3 Proposal payload

The proposal payload shall be the same as RFC 2408, section 3.5. The KDC expects the SPI Size to be zero, but if the SPI Size is non-zero, the contents of the SPI shall be ignored. The KDC shall support multiple transforms.

9.1.3.3.4 Transform payload

The transform payload shall be the same as RFC 2408, section 3.6. Required SA attributes are defined in RFC 2409 section 4 and copied below:

- Encryption Algorithm, value 1
  – Key Length, value 14. Required if the encryption algorithm does not specify a key length (i.e. AES_CBC). It is omitted if the encryption algorithm has an implied length (i.e. 3DES_CBC)
- Hash Algorithm, value 2
- Authentication Method, value 3
- Group Description, value 4

Optional SA attributes that may be supported by the KDC are:

- Life Type, value 11
- Life Duration, value 12

The domain of each attribute is defined in Table 1. A transform payload with additional attributes shall result in the KDC rejecting that payload.

9.1.3.4 Key exchange (2)

Once a security association is negotiated, the GM and KDC shall be able to exchange Diffie-Hellman (DH) public values based on the DH group description agreed upon in the SA. The key exchange sequence is shown in Figure 22.

```
        Initiator (GM)                       Responder (KDC)
      ----------------                     ------------------
      (1) HDR, SA                    -->
                                     <--   HDR, SA
      (2) HDR, KE, Ni                -->
                                     <--   HDR, KE, Nr
      (3) HDR*, IDii, CERT, SIG_I    -->
                                     <--   HDR*, IDir, CERT, SIG_R
```

IEC

Figure 22 – IKEv1 main mode exchange: key exchange messages

In this key exchange sequence, the KDC shall receive an ISAKMP message with a Key Exchange (KE) payload and a nonce (Ni) payload. The KE payload is the GM's DH public value and the Ni payload is the GM's nonce. Given the GM's DH public value and Ni, the secret and derived keys can be calculated.

As stated in the RFC 2409, section 5, the nonce shall be between 8 and 256 bytes. The KDC shall create a nonce that is at least half the size of the hash block size.

The KE and Ni payloads shall be the same as defined in RFC 2408, sections 3.8 and 3.14 respectively.

9.1.3.5 ID authentication exchange (3)

9.1.3.5.1 General

Once the connection has been secured, the last exchange of security messages performs mutual authentication. The ID authentication sequence is shown in Figure 23.

```
        Initiator (GM)                       Responder (KDC)
      ----------------                     ------------------
      (1) HDR, SA                    -->
                                     <--   HDR, SA
      (2) HDR, KE, Ni                -->
                                     <--   HDR, KE, Nr
      (3) HDR*, IDii, CERT, SIG_I    -->
                                     <--   HDR*, IDir, CERT, SIG_R
```

IEC

Figure 23 – IKEv1 Main Mode Exchange: ID authentication messages

The last message received from the GM shall contain the GM's ISAKMP Identification (Identification payload), its Identity Certificate (Certificate payload), and the signature of HASH_I using its Identity X.509 Certificate (Signature payload).

9.1.3.5.2 Identification payload

The ID Payload (IDii, IDir) shall be the same as defined in RFC 2408 section 3.8. The ID Types are technically DOI specific. GDOI RFC 6407 does not specify its own types but implies that the types defined for IPSec DOI are used in GDOI. These types are maintained by IANA under IKEv2 (RFC 5996 [41]). Therefore, the ID Type supported shall be ID_DER_ASN1_DN, which has a value of 9. The contents of the payload shall be the DER encoded Subject ID from the GM's X.509 certificate. All other ID types shall not be supported and shall result in an error.

9.1.3.5.3 Certificate payload

The certificate payload (CERT) shall contain the DER encoded X.509 Identity Certificate and shall be as defined in RFC 2408, section 3.9. The *X.509 Certificates – Signature* Certificate Encoding Type that has a value of 4 shall be supported. All other encoding types shall not be supported and shall result in an error.

Although RFC 2409 specifies that one or more certificate payloads may be optionally included, the KDC shall always include one and only one certificate payload in the ISAKMP message.

9.1.3.5.4 Signature payload

The signature payload (SIG_I, SIG_R) shall be unmodified from its definition in RFC 2408 section 3.12. The content shall be the signature, which is generated by using the GM's certificate, of HASH_I defined in RFC 2409, section 5 and depicted in Figure 24.

```
HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b )
```

IEC

Figure 24 – IKEv1 HASH_I calculation

Refer to RFC 2409 section 3 for definition of the notations.

It is important to note that the RSA signature is not a standard PKCS #1 formatted signature, but instead is simply encrypted using the certificate's RSA private key as described in RFC 2409 section 5.1 – IKE Phase 1 Authenticated With Signatures:

> ***"Since the hash algorithm used is already known there is no need to encode its OID into the signature. In addition, there is no binding between the OIDs used for RSA signatures in PKCS #1 and those used in this document. Therefore, RSA signatures MUST be encoded as a private key encryption in PKCS #1 format and not as a signature in PKCS #1 format (which includes the OID of the hash algorithm)."***

9.1.4 Phase 1/2 ISAKMP informational exchange type 5

9.1.4.1 General

The KDC shall use ISAKMP informational exchanges to notify its peers that an error has occurred. The KDC shall not use ISAKMP informational exchanges to provide status or delete SAs as defined in IKEv1 RFC 2409, section 5.7.

An informational exchange may be sent during either a phase 1 or phase 2 exchange. A phase 1 informational exchange is not encrypted and has the message ID field in the ISAKMP header set to zero. A phase 2 informational exchange is encrypted and has a unique message ID. A unique message ID shall be provided so that an applicable cryptographic initialization vector (IV) may be calculated.

9.1.4.2 Phase 1 informational exchange

9.1.4.2.1 General

A phase 1 Informational exchange may be initiated while the ISAKMP connection is being established to indicate that an error in the phase 1 exchange has occurred. The phase 1 Information exchange is shown in Figure 25.

```
Initiator                          Responder
-----------                        -----------
HDR*, N              -->
```

IEC

Figure 25 – Phase 1 Informational Exchange

Both the GM and KDC may be the initiator of a phase 1 informational exchange. The informational exchange message shall have a single notification payload (N) as defined in RFC 2408, section 3.14.

The Delete payload defined in RFC 2408, section 3.15 shall not be supported by the KDC for a phase 1 Informational exchange. Therefore, the KDC shall not send an informational exchange with a Delete payload and shall ignore any Delete payloads in a received informational exchange.

Additionally, the Hash payload shall not be supported for a phase 1 informational exchange. Therefore, the KDC shall not send a phase 1 informational exchange with a Hash payload and shall ignore any Hash payloads received in a phase 1 informational exchange. The phase 1 informational exchange message shall have the Message ID field in the ISAKMP header set to 0 and shall not be encrypted.

9.1.4.2.2 Notification payload

The Notification payload is defined in RFC 2408, section 3.14. For a phase 1 informational exchange, the payload field values are defined as:

1) Domain of Interpretation – Value of 2, GDOI
2) Protocol-ID – Value of 0
3) SPI Size – Value of 0. The I-Cookie and R-Cookie are used as the SPI
4) Notify Message Type – Any value defined in RFC 2408, section 3.14.1
5) SPI – not included in payload
6) Notification Data – not included in payload.

9.1.4.3 Phase 2 informational exchange

GDOI RFC 6407 mentions of an Informational exchange are in reference to a GM not accepting an SA policy. If the policy in the SA payload is not acceptable to the GM, “the GM should tear down the Phase 1 session after notifying the KDC with an ISAKMP Informational Exchange containing a Delete payload” (RFC 6407, section 3.3).

However, there is no mention on how the KDC or GM should handle the case where the hash is incorrect or a GM does not have access privilege to the group. To cover for these cases, this standard adds the requirement that a KDC shall embed a Notification Payload in the GROUPKEY-PULL exchange itself. If a failure occurs while processing a GROUPKEY-PULL message from the GM, the KDC shall return a GROUPKEY-PULL message on the same exchange with a single Notification Payload indicating the error code. The GROUPKEY-PULL exchange shall be terminated.

The KDC shall support receiving both a Notification Payload embedded in the GROUPKEY-PULL exchange and receiving an Informational Exchange with a Delete payload. If the SPI of the Notification/Delete payload matches an SA of an existing GROUPKEY-PULL exchange, it shall stop the exchange.

9.1.5 Phase 2 GDOI GROUPKEY-PULL exchange type 32

9.1.5.1 General

GDOI (RFC 6407) defines the Phase 2 GROUPKEY-PULL exchange to allow Group Members a way to pull the shared security associations for a single secure multicast group. The GROUPKEY-PULL exchange is sent on the IKE SA after the connection is secured with the phase 1 IKEv1 main mode exchange. RFC 6407, section 3.2 defines the GROUPKEY-PULL exchange in Figure 26.

```
     Group Member                    KDC
     ------------                    ----
(1)  HDR*, HASH(1), Ni, ID     -->
(2)                            <--   HDR*, HASH(2), Nr, SA
(3)  HDR*, HASH(3) [,GAP]      -->
(4)                            <--   HDR*, HASH(4), [SEQ,] KD

     * Protected by the Phase 1 SA; encryption occurs after HDR
                                                             IEC
```

Figure 26 – GD004FI GROUPKEY-PULL as define in RFC 6407

Refer to RFC 2409, section 3.2 and RFC 6407, section 1.3 for notation, acronyms and abbreviations.

The Group Member shall always initiate a GROUPKEY-PULL exchange. The KDC shall always be the responder of a GROUPKEY-PULL exchange.

When the KDC receives the initial GROUPKEY-PULL exchange message from the GM, the message is validated and a new GROUPKEY-PULL exchange is started. The payloads are extracted and the HASH(1) is calculated and validated. An incorrect hash calculation shall result in the exchange failing and a Phase 2 informational exchange shall be initiated by the KDC indicating an error type of INVALID_HASH_INFORMATION (value 23).

An incorrect hash calculation shall result in the exchange failing and a GROUPKEY-PULL response message shall be sent back to the GM with a Notification payload indicating an error type of INVALID_HASH_INFORMATION (value 23).

The secure multicast group ID shall be extracted and if the secure multicast group is not found or the GM is not an authorized group member, the exchange shall fail and a GROUPKEY-PULL response message shall be sent back to the GM with a Notification payload indicating error INVALID-ID-INFORMATION (value 18) and AUTHENTICATION_FAILURE (value 24) respectively.

9.1.5.2 Hash computations

The hashes computed for the GROUPKEY-PULL exchange shall use the secure hash algorithm from the phase 1 SA. The hashes shall be secured with the phase 1 authentication secret, SKEYID_a, as defined in RFC 2409, section 5. The different hashes used in the GROUPKEY-PULL exchange shall be computed over the information specified in Figure 27.

```
HASH(1) = prf(SKEYID_a, M-ID | Ni  | ID)
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | Nr  | SA)
HASH(3) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [ | GAP ])
HASH(4) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [ | SEQ ] | KD)
```

IEC

Figure 27 – GROUPKEY-PULL hash computations

9.1.5.3 Multi-sender and counter mode encryption algorithm

IEC 61850 DATA SAs shall support AES-GCM and AES-GMAC counter mode algorithms. Counter mode algorithms provide the capability for there to be multiple senders over a single SA. This is achieved by ensuring that each sender use unique initialization vectors (IV) for each packet sent. GDOI RFC 6407, section 3.5 specifies that the KDC implement RFC 6054 in order to assign a portion of the IV space to each sender in the group[3].

Support for the request of Sender IDs from a GM for counter mode based IEC 61850 key groups is not required. If a GAP payload is received from a GM with a Sender-ID GAP attribute (value 3), the KDC shall fail the GROUPKEY-PULL exchange. An error type of ATTRIBUTES-NOT-SUPPORTED (value 13) shall be returned in an Informational exchange.

NOTE It is to be determined whether multiple senders on a single DATA SA will be supported in a future edition of this standard.

9.1.5.4 SA-KEK, SEQ, KEK/LKH key download payload support (rekey via GROUPKEY-PUSH)

Since the GROUPKEY-PUSH is optional, the GROUPKEY-PULL exchange is not required to support an SA-KEK payload, SEQ payload, or KEK/LKH Key Packets in the Key Download payload.

The GROUPKEY-PUSH exchange is expected be supported in a future edition of this standard.

NOTE It is to be determined when the GROUPKEY-PUSH exchange will be supported by IEC 61850 security protocols.

9.1.5.5 GROUPKEY-PULL group SA request exchange

9.1.5.5.1 General

The initial SA Request exchange is shown in Figure 28.

3 RFC 6054 – Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic (http://tools.ietf.org/html/rfc6054).

```
    Group Member                      GKDC
    ------------                      ----
(1) HDR*, HASH(1), Ni, ID      -->
(2)                            <--    HDR*, HASH(2), Nr, SA
(3) HDR*, HASH(3) [,GAP]       -->
(4)                            <--    HDR*, HASH(4), [SEQ,] KD
```

IEC

Figure 28 – GROUPKEY-PULL initial SA request exchange

The GM shall initiate a GROUPKEY-PULL exchange by sending message (1) to the KDC. The HASH (1) and Ni payloads are defined in RFC 6407, section 3. The ID Payload has been extended to support IEC 61850 secure multicast groups.

9.1.5.5.2 Identification payload

9.1.5.5.2.1 General

The ID payload (ID) for the phase 2 exchange is the same ID payload used in the phase 1 exchange (see 9.1.3.5.2). The difference is that in the phase 1 exchange, the Identification Data contained is that of the Group Member, while for the phase 2 exchange the ID identifies the Key Group for the SAs to be pulled as shown in Figure 29. The KDC supports IEC 61850 secure multicast groups.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!    ID Type    !         DOI-Specific ID Data = 0              !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
~                      Identification Data                      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

IEC

Figure 29 – RFC 6407 Identification Payload

The format of the identification payload consists of:

[1] **Next Payload** – see RFC 2408 for definition and use.

[2] **Reserved** – see RFC 2408 for definition and use.

[3] **Payload Length** – see RFC 2408 for definition and use.

[4] **ID Type** – An IEC 61850 secure multicast group is identified by an ID Type of ID_OID defined in the IETF Internet Draft GDOI Protocol Support for IEC 62351 [52] section 2.1. The ID_OID ID Type has a value of 13. The Identification Data for ID_OID represents an ASN.1 Object ID (OID)[4] and its OID specific data. Both the OID and the OID data are encoded using DER encoding rules[5]. The field definitions within the Identification Data payload are defined in and depicted in Figure 30. For example, an OID could represent a GOOSE or Sampled Value protocol, also in other cases

[4] The Object ID format is defined in ITU-T-X.683 – http://www.itu.int/ITU-T/studygroups/com17/languages/X.683-0207.pdf.

[5] Distinguished Encoding Rules is define in ITU-T-X.690 – http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf.

IEC 61850 also specifies a particular multicast destination address to be described in the OID Specific Payload field.

[5] **DOI-Specific ID Data** – Shall be set to 0.

[6] **Identification Data** – is ID_OID Specific and described in the following paragraph.

9.1.5.5.2.2 Identification data (IEC 61850 objects)

The GDOI Identification Payload (e.g. Phase 2 request) in a GDOI GROUPKEY-PULL exchange allows the Group Member (GM) to declare the group it would like to join. A group is defined by an ID payload as defined in GDOI RFC 6407 ([44]). Figure 30 has been extracted from the IETF Internet Draft [52] and defines a specific ID Type value and format of the Identification Data.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!  OID Length    !                        OID                   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!  OID Specific Payload Length  !     OID Specific Payload      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

IEC

Figure 30 – ID_OID Identification Data

[1] **OID Length**: This length shall be an unsigned integer value and shall specify the number of octets of the ASN.1 encoded OID, whose value follows the length;

[2] **OID**: This set of octets represents an ASN.1 encoded Object Identifier. The value of the identifier defines the payload that follows. It is the use of this Object Identifier that will allow other organizations or standards to utilize this payload extension process without definition collision. The Object Identifier values are defined as mandatory (m) or optional (o) in Table 2.

[3] **OID Specific Payload Length** (2 octets) – Length of the OID Specific Payload. Set to zero if the OID does not require an OID Specific Payload.

[4] **OID Specific Payload** (variable) – OID specific selector encoded in DER. If OID Specific Payload Length is set to zero this field does not appear in the ID payload.

At the culmination of a GROUPKEY-PULL exchange, the key server will have sent group policy to all authorized group members, allowing receiving group members to participate in secure group communications.

The KDC shall support the mandatory Object IDs defined in Table 2, which identify the IEC 61850 stream that the GM is requesting Key material for.

Table 2 – IEC 61850 Object IDs: Mandatory (m) and Optional (o)

Object Identifier Name	Description	Value	m/o
61850_ETHERNET_GOOSE	Specifies that the payload is requesting a key for an IEC 61850-8-1 GOOSE APDU.	1.0.62351.9.61850.8.1.1	o
61850_UDP_ADDR_GOOSE	Specifies that the payload is requesting a key for a routable GOOSE APDU that is being sent to a particular destination IP address.	1.0.62351.9.61850.8.1.2	m
61850_UDP _Tunnel	Specifies that the payload is requesting a key for an IEC 61850 routable Tunnel APDU that is being sent to a particular destination IP address.	1.0.62351.9.61850.8.1.4	m
61850_ETHERNET_SV	Specifies that the payload is requesting a key for an IEC 61850-9-2 SV APDU.	1.0.62351.9.61850.9.2.1	o

Object Identifier Name	Description	Value	m/o
61850_UDP_ADDR_SV	Specifies that the payload is requesting a key for a routable IEC 61850 SV APDU.	1.0.62351.9.61850.9.2.2	m
61850_IP_ISO9506	Specifies that the payload is requesting a key for an IEC 61850-8-1 ISO 9506 endpoint. This payload definition is out-of-scope.	1.0.62351.9.61850.8.1.4	o

NOTE That the utilized OIDs have been defined in the IEC 62351 space starting with 1.0.62351.9.xxx as opposed to the OIDs used in IEC 61850-90-5 starting with 1.2.840.10070.xxx

9.1.5.5.2.3 OID specific payloads

The payloads for the UDP versions of GOOSE (61850_UDP_ADDR_GOOSE) and SV (61850_UDP_ADDR_SV) OIDs[6] are the same. The ASN.1 Sequence for 61850_UDP_ADDR_GOOSE and 61850_UDP_ADDR_SV OID specific data are specified in Figure 31.

```
IecUdpAddrPayload ::= SEQUENCE
{
        version     INTEGER { v1(1) },
        ipAddress   IPADDRESS,
        dsRef       VisibleString(SIZE(1..128))
}
```

IEC

Figure 31 – 61850_UDP_ADDR_GOOSE/SV ASN.1 BNF

[1] **Version** is a single octet value that represents the version of the particular payload. Unless otherwise specified, the value of the VERSION shall be one (1).

[2] **IPADDRESS** is a value component that allows the use of either an IPv4 or IPv6 destination address for the entity associated with the key being requested. The IPADDRESS ASN.1 Sequence is specified in Figure 32.

[3] The dsRef value component allows for the specification of an IEC 61850 DataSet Reference (DSRef), as specified in IEC 61850-7-2.

```
IPADDRESS ::= SEQUENCE
{
   typeOfAddress ENUMERATED { IPv4(0), IPv6(1) },
   address  CHOICE {
      ip     OCTECT STRING (SIZE(4 | 16)),
      dns    VisibleString (SIZE(1..65536))
      }
}
```

IEC

Figure 32 – IPADDRESS ASN.1 BNF

[6] The payload for the IP versions of GOOSE and SV are the same.

Figure 33 is an example of the 61850 OID Address Payload for either 61850_UDP_ADDR_GOOSE or 61850_UDP_ADDR_SV OIDs.

```
groupaddr IecUdpAddrPayload ::=
{
   version v1,
   ipAddress
   {
        typeOfAddress IPv4,
        address dns: "www.iec.org"
   }
   dsRef "@somedataref"
}

DER Encoding:

30230201 0130100A 01001A0B 7777772E
6965632E 6F72671A 0C40736F 6D656461
74617265 66
```

IEC

Figure 33 – Example IecUdpAddrPayload ASN.1 Data with DER Encoding

The ASN.1 Sequence for 61850_UDP_TUNNEL OID specific data is specified in Figure 34.

```
IecUdpTunnelPayload ::= SEQUENCE
{
    version      INTEGER { v1(1) },
    ipAddress    IPADDRESS
}
```

IEC

Figure 34 – 61850_UDP_TUNNEL Payload ASN.1 BNF

NOTE The dsRef field is not present in this payload because multiple Ethernet multicast frames (e.g. GOOSE or SV) may be sent in one Tunnel SPDU. Therefore, the destination IP address itself differentiates the data sets.

The OID payload for the Ethernet versions of GOOSE (61850_ETHERNET_GOOSE) and SV (61850_ETHERNET_SV) are the same. The ASN.1 Sequence for 61850_ETHERNET_GOOSE/SV OID specific data is specified in Figure 35.

```
IecEthernetAddrPayload ::= SEQUENCE
{
    version  INTEGER { v1(1) },
    dstMAC   OCTET STRING (SIZE(6)),
    dsRef    VisibleString (SIZE(1..256))
}
```

IEC

Figure 35 – 61850_ETHERNET_GOOSE/SV Payload ASN.1 BNF

[1] Version is a single octet value that represents the version of the particular payload. Unless otherwise specified, the value of the VERSION shall be one (1).

[2] The dstMAC is a value that consists of six (6) octets. The value shall be in Ethernet transmission order.

[3] The dsRef value component allows the specification of an IEC 61850 DataSet Reference (DSRef), as specified in IEC 61850-7-2.

9.1.5.6 SA TEK payload

The SA TEK payload shall contain security attributes for a single set of policies associated with a group TEK. The type of policy to be used with the TEK is described by a Protocol-ID field included in the SA TEK. As shown in Figure 36 reproduced from RFC 6407, each Protocol-ID describes a particular TEK Protocol-Specific Payload definition.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 ! Next Payload  !   RESERVED    !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 ! Protocol-ID   !       TEK Protocol-Specific Payload           ~
 +-+-+-+-+-+-+-+-+                                               ~
 ~                                                               ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

IEC

Figure 36 – RFC 6407 SA TEK Payload

The KDC shall comply with the IEC-61850 SA TEK payload format defined in IETF draft GDOI Protocol Support for IEC 62351 [52], section 2.2. The IEC-61850 SA TEK payload defines two optional SA Data Attributes, the Activation Time Delay (SA_ATD) and the Key Delivery Assurance (SA_KDA) attributes. The values for these attributes are defined in draft-weis-gdoi-iec62351-9-05 section 4.0, IANA Considerations.

The Protocol ID name of GDOI_PROTO_IEC_61850 is marked as TBD1 in the IETF draft GDOI Protocol Support for IEC 62351 [52], section 2.2. The protocol ID shall have a value of decimal 161, which is in the domain of private use values.

9.1.5.7 IEC-61850 SA TEK payload

GDOI_PROTO_IEC_61850 SA TEK shall include an OID and (optionally) an OID Specific Payload that together define the selectors for the network traffic. The selector fields shall be followed by security policy fields indicating how the specified traffic is to be protected. As shown in Figure 37 reproduced from [52] each IEC 61850 TEK Payload describes a particular TEK Protocol-Specific Payload definition.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !  OID Length   !                     OID                        ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !  OID Specific Payload Length  !     OID Specific Payload      ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !                              SPI                              !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !           Auth Alg            !            Enc Alg            !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !                   Remaining Lifetime Value                    !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
 !                     SA Data Attributes                        ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
                                                              IEC
```

Figure 37 – IEC-61850 SA TEK Payload

The GDOI_PROTO_IEC_61850 SA TEK Payload fields are defined in [52] as follows:

a) OID Length (1 octet) – Length of the OID field.

b) OID (variable) – An ASN.1 object Identifier encoded using DER. OIDs defined in IEC 61850 declare the type of IEC 61850 message to be protected, as defined in Table 2.

c) OID Specific Payload Length (2 octets) – Length of the OID Specific Payload. This field is set to zero if the policy does not include an OID Specific Payload.

d) OID Specific Payload (variable) – The traffic selector (e.g., multicast address) specific to the OID encoded using DER. Some OID policy settings do not require the use of an OID Specific Payload, in which case this field is not included in the TEK and the OID Specific Payload Length is set to zero.

e) SPI (4 octets) – Identifier for the Current Key. This field represents a SPI.

f) Auth Alg (2 octets) – Authentication Algorithm ID. Valid values are defined in [52], Section 2.2.2. HMAC-SHA256-128 is mandatory.

g) Enc Alg (2 octets) – Confidentiality Algorithm ID. Valid values are defined in [52] Section 2.2.3. AES-CBC-128 is mandatory.

h) Remaining Lifetime value (4 octets) – The number of seconds remaining before this TEK expires. A value of zero (0) shall indicate that the TEK does not have an expiry time. Maximum lifetime shall be correlated with the CRL refresh time to ensure that group members with expired certificates or revoked certificates are handled according to the organization’s security policy.

i) SA Data Attributes (variable length) – Contains zero or more attributes associated with this SA, [52], section 2.2.4 defines attributes.

There are some restrictions on how the Authentication Algorithm and the Confidentiality Algorithm can be combined. The restrictions are summarized below:

a) AES-GCM/GMAC is a combined cipher providing both encryption and authentication. If AES-GCM is specified for the confidentiality algorithm, the Authentication Algorithm field shall be set to RESERVED (value 1).

b) If AES-GCM is not specified for the confidentiality algorithm, an authentication algorithm shall be specified and the Authentication Algorithm field shall not be set to RESERVED (value 1).

c) A confidentiality algorithm is optional and the Confidentiality Algorithm field may be set to NONE (value 1).

9.1.5.8 SPI discussion

RFC 6407 requires that characteristics of a SPI must be defined. A SPI in a GDOI_PROTO_IEC_61850 SA TEK is represented as a Key Identifier (KeyID). The SPI size is 4 octets. The SPI is unilaterally chosen by the KDC using any method chosen by the implementation. However, an implementation needs to take care not to duplicate a SPI value that is currently in use for a particular group.

9.1.5.9 SA data attributes

9.1.5.9.1 General

The following attributes shall be present in the SA TEK for IEC 61850 SAs. The attributes shall follow the format described in [52], Appendix C.

9.1.5.9.2 Activation time delay (SA_ATD) SA data attribute (value 1)

A KDC shall distribute an SA TEK in advance of when it is expected to be used. This is communicated to group members using the SA Activation Time Delay (SA_ATD) attribute. When a GM receives an SA_TEK with this attribute, it shall wait for the number of seconds contained within the attribute before installing it for either transmission or receiving. The KDC shall include the SA_ATD SA data attribute, even if the value is 0. The SA_ATD attribute is assigned a value of 1.

9.1.5.9.3 Key delivery assurance (SA_KDA) SA data attribute (value 2)

Group policy may include notifying a multicast source ("Publisher") of an indication of whether multicast receivers ("Subscribers") have previously received the SA TEK. This notification allows a Publisher to set a policy as to whether to activate the new SA TEK or not based on the percentage of Subscribers that are able to receive packets protected by the SA TEK. The attribute value is a number between 0 and 100 (inclusive). Support for KDA is optional. If KDA is not supported, the KDC shall set the SA_KDA value to 100.

The SA_KDA attribute is assigned a value of 2.

9.1.6 GROUPKEY-PULL group key download exchange

After the SA Policies are sent to the GM, the KDC shall send the key material for each SA. The exchange is described in Figure 38.

```
     Group Member                      KDC
     ------------                      ----
 (1) HDR*, HASH(1), Ni, ID     -->
 (2)                           <--     HDR*, HASH(2), Nr, SA
 (3) HDR*, HASH(3)             -->
 (4)                           <--     HDR*, HASH(4), KD
                                                          IEC
```

Figure 38 – GROUPKEY-PULL Key Download Exchange

Before the key material is sent to the GM, the GM shall first respond to the SA exchange by sending message (3) with the Hash payload containing the HASH(3) computation shown here:

```
HASH(3) = prf(SKEYID_a, M-ID | Ni_b | Nr_b )
```

If the KDC cannot validate the hash calculation, a phase 2 informational exchange shall be initiated with an error indicating INVALID_HASH_INFORMATION (value 23).