International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1365

(03/2020)

SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

Secure applications and services (2) – Internet of things (IoT) security

Security methodology for the use of identity-based cryptography in support of Internet of things (IoT) services over telecommunication networks

Recommendation ITU-T X.1365

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

PUBLIC DATA NETWORKS	X.1–X.199
OPEN SYSTEMS INTERCONNECTION	X.200–X.299
INTERWORKING BETWEEN NETWORKS	X.300–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	X.600–X.699
OSI MANAGEMENT	X.700–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	X.850–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999
INFORMATION AND NETWORK SECURITY	
General security aspects	X.1000–X.1029
Network security	X.1030–X.1049
Security management	X.1050–X.1069
Telebiometrics	X.1080–X.1099
SECURE APPLICATIONS AND SERVICES (1)	
Multicast security	X.1100–X.1109
Home network security	X.1110–X.1119
Mobile security	X.1120–X.1139
Web security	X.1140–X.1149
Security protocols (1)	X.1150–X.1159
Peer-to-peer security	X.1160–X.1169
Networked ID security	X.1170–X.1179
IPTV security	X.1180–X.1199
CYBERSPACE SECURITY	
Cybersecurity	X.1200–X.1229
Countering spam	X.1230–X.1249
Identity management	X.1250–X.1279
SECURE APPLICATIONS AND SERVICES (2)	
Emergency communications	X.1300–X.1309
Ubiquitous sensor network security	X.1310–X.1319
Smart grid security	X.1330–X.1339
Certified mail	X.1340–X.1349
Internet of things (IoT) security	**X.1360–X.1369**
Intelligent transportation system (ITS) security	X.1370–X.1389
Distributed ledger technology security	X.1400–X.1429
Distributed ledger technology security	X.1430–X.1449
Security protocols (2)	X.1450–X.1459
CYBERSECURITY INFORMATION EXCHANGE	
Overview of cybersecurity	X.1500–X.1519
Vulnerability/state exchange	X.1520–X.1539
Event/incident/heuristics exchange	X.1540–X.1549
Exchange of policies	X.1550–X.1559
Heuristics and information request	X.1560–X.1569
Identification and discovery	X.1570–X.1579
Assured exchange	X.1580–X.1589
CLOUD COMPUTING SECURITY	
Overview of cloud computing security	X.1600–X.1601
Cloud computing security design	X.1602–X.1639
Cloud computing security best practices and guidelines	X.1640–X.1659
Cloud computing security implementation	X.1660–X.1679
Other cloud computing security	X.1680–X.1699
QUANTUM COMMUNICATION	X.1700–X.1729

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T X.1365

Security methodology for the use of identity-based cryptography in support of Internet of things (IoT) services over telecommunication networks

Summary

Recommendation ITU-T X.1365 provides a security methodology for the use of identity-based cryptography (IBC) public key technology in support of Internet of things (IoT) services over telecommunication networks, including mechanisms of identity management, key management architecture, key management operations and authentication.

Traditional certificate-based security methodology involves heavyweight key management operations including certificate issuance, querying and revocation. Such systems face great difficulty in keeping up with the increasing numbers of devices connected in IoT while maintaining adequate performance.

IBC technology is another security methodology that uses an entity's identity as a public key. An essential feature of IoT is that everything has a unique identifier (ID). Using such IDs as public keys has the benefit of no certificates being required. Consequently, an IBC security solution utilizes simpler key management, enables distributed authorities to control their own devices and scales well to both a high number of endpoints and diverse devices.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T X.1365	2020-03-26	17	11.1002/1000/14089

Keywords

IoT, IBC, identity-based cryptography, security methodology, user data security.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendations unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-Ts purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T X.1365

Security methodology for the use of identity-based cryptography in support of Internet of things (IoT) services over telecommunication networks

1 Scope

This Recommendation specifies a security methodology for the use of identity-based cryptography (IBC) technology in support of Internet of things (IoT) services over telecommunications networks. This security methodology includes mechanisms for device identification, private key issue, public parameter lookup and authentication protocols.

NOTE – This methodology is not limited to IoT service, it can be used by other services also.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[IETF RFC 4764] IETF RFC 4764 (2007), *The EAP-PSK protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method*.

[IETF RFC 5091] IETF RFC 5091 (2007), *Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems*.

[IETF RFC 5216] IETF RFC 5216 (2008), *The EAP-TLS Authentication Protocol*.

[IETF RFC 5280] IETF RFC 5280 (2008), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

[IETF RFC 5408] IETF RFC 5408 (2009), *Identity-Based Encryption Architecture and Supporting Data Structures*.

[IETF RFC 5480] IETF RFC 5480 (2009), *Elliptic Curve Cryptography Subject Public Key Information*.

[IETF RFC 5958] IETF RFC 5958(2010), *Asymmetric Key Packages*.

[IETF RFC 6507] IETF RFC 6507 (2012), *Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)*.

[IETF RFC 6508] IETF RFC 6508 (2012), *Sakai–Kasahara Key Encryption (SAKKE)*.

[IETF RFC 6960] IETF RFC 6960 (2013), *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*.

[IETF RFC 7250] IETF RFC 7250 (2014), *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*.

[IETF RFC 8446] IETF RFC 8446 (2018), *The Transport Layer Security (TLS) Protocol Version 1.3*.

[ISO/IEC 11770-3] ISO/IEC 11770-3:2015, *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*.

[ISO/IEC 14888-3] ISO/IEC 14888-3:2018, *IT Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*.

[ISO/IEC 18033-5] ISO/IEC 18033-5:2015, *Information technology – Security techniques – Encryption algorithms – Part 5: Identity-based ciphers*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 identity provider [b-ITU-T Y.2720]: An entity that creates, maintains and manages trusted identity information of other entities (e.g., users/subscribers, organizations, and devices) and offers identity-based services based on trust, business and other types of relationship.

3.1.2 identifier (ID) [b-ITU-T E.101]: A series of digits, characters and symbols used to identify uniquely a subscriber, a user, a network element, a function, a network entity, a service or an application. Identifiers can be used for registration or authorization. They can be either public to all networks or private to a specific network (private IDs are normally not disclosed to third parties).

3.1.4 master public key (MPK) [ISO/IEC 18033-5]: Public value uniquely determined by the corresponding master secret key.

3.1.3 master secret key (MSK) [ISO/IEC 18033-5]: Secret value used by the private key generator to compute private keys for an IBE algorithm.

3.1.5 private key generator (PKG) [ISO/IEC 18033-5]: Entity or function which generates a set of private keys.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 identity domain: A collection of entities that share the same set of public parameters and identity naming rules.

3.2.2 public parameter: One of the parameters for cryptographic computation, including a selection of a particular cryptographic scheme or function from a family of cryptographic schemes or functions, or from a family of mathematical spaces and the master public key.

3.2.3 public parameter server: An entity that provides public parameters upon request.

3.2.4 security module (SecM): A piece of software or hardware or the composition of software and hardware that securely implements cryptographic mechanisms and provides security services.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

4G	fourth Generation
5G	fifth Generation
AuC	Authentication Centre
AGW	Aggregate Gateway
AK	Authentication Key
AKA	Authenticated Key Agreement
AN	Access Node

AS	Access System
ASN.1	Abstract Syntax Notation one
AU	Authentication Unit
BN	Barreto-Naehrig
BLS-12	Barreto-Lynn-Scott embedding degree 12
BLS-24	Barreto-Lynn-Scott embedding degree 24
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules
EAP	Extensible Authentication Protocol
ECCSI	Elliptic Curve-based Certificateless Signatures for Identity-based encryption
EID	eUICC-ID
EIS	eUICC Information Set
eUICC	Embedded Universal Integrated Circuit Card
EUM	eUICC Manufacturer
GW	Gateway
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
IBAKA	Identity-Based Authenticated Key Agreement
IBC	Identity-Based Cryptography
IBE	Identity-Based Encryption
IBS	Identity-Based Signature
ID	Identifier
IdP	Identity Provider
IMSI	International Mobile Subscription Identity
IoT	Internet of Things
ISP	IoT Service Platform
IRL	Identity Revocation List
ISD	Issuer Security Domain
KDF	Key Derivation Function
KDK	Key Derivation Key
KEK	Key Encryption Key
KEM	Key Encapsulation Mechanism
KMIP	Key Management Interoperability Protocol
KMS	Key Management Service
KPAK	KMS Public Authentication Key
KSS-16	Kachisa-Schaefer-Scott embedding degree 16
KSS-18	Kachisa-Schaefer-Scott embedding degree 18

LTE	Long-Term Evolution
LTE-M	Long-Term Evolution, category M1
MAC	Media Access Control
MNO	Mobile Network Operator
MSK	Master Secret Key
NB-IoT	Narrowband Internet of Things
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OISP	Online Identity Status Protocol
PKG	Private Key Generator
PKI	Public Key Infrastructure
PPS	Public Parameter Server
PVT	Public Verification Token
RSF	Revocation Server Function
SecM	Security Module
SK	Sakai-Kasahara
SM-DP	Subscription Manager Data Preparation
SM-SR	Subscription Manager Secure Routing
SOK	Sakai-Ohgishi-Kasahara
SSK	Secret Signing Key
TLS	Transport Layer Security
TLV	Tag, Length and Vector
TVP	Time-Variant Parameter
UE	User Equipment
UICC	Universal Integrated Circuit Card

5 Conventions

None.

6 Overview

The IoT can be viewed as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies (ICT) according to clause 6.1 of [b-ITU-T Y.4000]. Security of IoT is one of the foremost concerns due to the ubiquitous nature of devices coupled with increasing sensitivity of user data. [b-ITU-T Y.4100] describes high-level common security requirements of IoT, including communication security, data management security, service provision security, as well as mutual authentication and authorization. [b-ITU-T X.1361] further analyses security threats and challenges in an IoT environment and describes capabilities that could address and mitigate them. The necessary security capabilities defined [b-ITU-T X.1361] include:

– a secure communication capability for supporting secure, trusted and privacy protected communication;
– a secure key management capability for supporting secure communications;
– a secure data management capability for providing secure, trusted and privacy protected data management;
– an authentication capability for authenticating devices;
– an authorization (access control) capability for authorizing devices;
– a capability to implement secure protocols based on lightweight cryptographic algorithms.

IoT devices are characterized by a constraint of resources such as computation and communication capabilities. The nature of IoT devices brings new challenges to satisfy security requirements in an IoT system. Particularly, easy deployment, lightweight management operations and distributed authority are among the key factors when considering security solutions for IoT.

As described in [b-ITU-T X.1361], authentication, access control, as well as data integrity and confidentiality are among essential services required to secure IoT. Both symmetric-key and public-key cryptography mechanics can be exploited to provide such services.

A symmetric-key based security solution is relatively simple. However, it does not fit well in peer-to-peer scenarios such as machine-to-machine applications in IoT without an online service acting as a trust broker or without pre-sharing a secret pairwise among devices. Cross-system secure communication is also complicated without exposing the user secret to peer parties.

The traditional certificate-based public key cryptography solution involves heavyweight key management operations, including certificate issue, querying, distribution, verification and revocation. Such systems face significant challenges in keeping up with the pace of increases in device numbers and functionality on IoT while maintaining decent performance. Overhead of exchanging certificates in security protocols also causes problems, particularly in narrowband Internet of things (NB-IoT) networks which have a small packet data unit.

Identity-based cryptography (IBC) is another type of technology, which uses an entity's identity as a public key. As one of the essential features of IoT, everything has a unique identifier (ID). By using such IDs as public keys, no certificates are required. Consequently, an IBC security solution utilizes simpler key management, enables distributed authorities to control their own devices and scales well to both high number of endpoints and diversity of devices. As no certificates are transmitted, security protocols can be conducted more efficiently.

In an IBC system, a trusted party called a key management service (KMS) is in charge of generating each entity’s private key. Before providing the key generation service, the KMS starts a system initialization process by invoking an **IBSetup** function which, given a security parameter, determines a set of system parameters and generates a master secret key (MSK) and a master public key (MPK). Note that the KMS has the same function as private key generator (PKG). Thus, for convenience of expression, KMS and PKG are used interchangeably in this Recommendation, and the combination of system parameters and the MPK is referred to as the public parameters. The KMS keeps the MSK strictly confidential and makes the public parameters publicly available. If necessary, the public parameters may be published by a dedicated service public parameter server (PPS).

A typical IBC security system may use a range of IBC mechanisms including identity-based encryption (IBE), identity-based signature (IBS) and identity-based authenticated key agreement (IBAKA) to provide various security services including data confidentiality, entity authentication and secure channel establishment. All of these IBC algorithms can be deemed as the composition of two sets of functions. One set consists of key generation functions, which generate identity-based public and private key pairs. The private key generation function (**IBExtract)** generates a private key from an ID, the MSK and the public parameters. The identity public key derivation function (**IBDerivate**) computes a public key from an ID and the public parameters. The other set of functions, e.g.,

encryption or decryption (**IBEnc/IBDec**), signing or verification (**IBSign/IBVerify**) and authenticated session key establishment protocol, uses the key pairs generated to complete corresponding cryptographic operations.

IBC technology has been standardized by various standards development organizations, including the International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Internet Engineering Task Force (IETF), Institution of Electrical and Electronics Engineers (IEEE), European Telecommunications Standards Institute (ETSI) and Standard Administration of China (SAC). A list of some relevant standards developed by these organizations is given in the Bibliography. OneM2M is also considering use of IBC technologies for IoT networks in Release 4, whose security analysis can be found in [b-ETSI TR 118 508].

This Recommendation describes a security framework for using IBC technology to provide security capabilities for IoT services over telecommunications networks. The framework covers aspects of identity management, key management architecture, key management operations and authentication, as well as key agreement protocols of using IBC.

7 System reference architecture for IoT services over telecommunications networks

This clause presents a general system reference architecture for IoT services over telecommunications networks. Figure 1 shows a conceptual system reference architecture for IoT services. The system consists of three domains: IoT device, access system (AS), and IoT service platform (ISP).

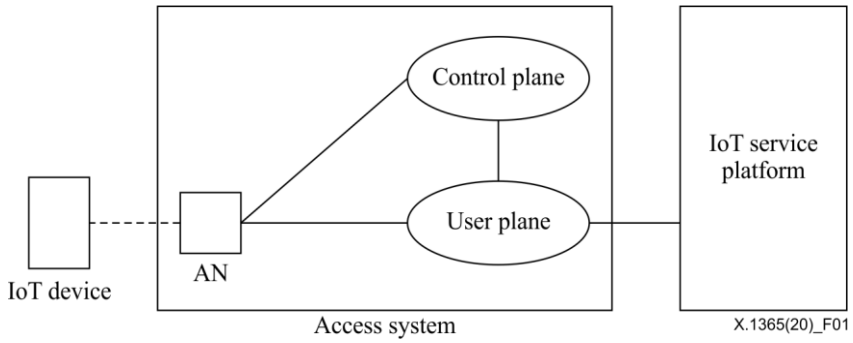

Figure 1 – Conceptual system architecture for Internet of things services

IoT devices are responsible for data collection or performing actions. Most IoT devices can establish a connection with a telecommunication system and communicate with an ISP. Nowadays, the majority of IoT devices connect to an ISP via a wireless link established with a telecommunication network. In this Recommendation, the AS refers to telecommunication networks. Usually it consists of two parts: access network (AN) and core network. The core network can be further subdivided into two parts: control plane and user plane, which are responsible for control signalling and data transmission, respectively.

The telecommunication network, as a traditional wireless connection, has already been used for several generations. Historically, telecommunication networks are designed to support mobile communication for human beings with seamless roaming features. In recent years, starting from fourth generation (4G) long-term evolution (LTE) networks, supporting of IoT devices are also considered in the design. For example, in 4G-LTE, category M1 (LTE-M) and NB-IoT technologies are developed to support IoT devices.

The majority of today's telecommunication systems consist of three components: terminals or user equipment (UE), an AN, and core networks. Here, it is assumed that both the AN and core networks belong to the AS, illustrated in Figure 1. IoT services are usually outside telecommunication networks with some interfaces for data transmission and service management. To provide better support for IoT services, telecommunication networks are including more IoT specific design in their system

specifications. Integration between telecommunications networks and IoT services has become closer in recent years.

With system specifications developed for fifth generation (5G) networks, public key technologies are supported for IoT services, including network access authentication. As mentioned in clause 6, compared to other public key technologies, IBC is simpler in management and more efficient in transmission. Therefore, the use of IBC for IoT services over telecommunication networks requires specification as a standard complementary to existing specifications.

8 Framework of using identity-based cryptography for IoT services over telecommunications networks

In this clause, a framework of using IBC public key technologies for IoT services in telecommunications networks is provided. The framework contains a system architecture by including necessary network components required when using IBC technologies. Furthermore, a key management framework for IBC is specified, as it is essential for a system using IBC technology. Other critical issues, such as key management, identity naming and authentication protocols, are also addressed in this framework.

8.1 IoT system architecture with identity-based cryptography

For IoT services run over telecommunications networks, IBC can be used for either network access authentication or service access authentication, or both. Network access authentication addresses whether a device is allowed to access the network, while service access authentication addresses whether a device can access an ISP.

IoT devices can access the telecommunication network directly or indirectly. Hence, there are two access models:

– direct-connected model: IoT devices connect to the AS directly;

– indirect-connected model: IoT devices connect to the AS via an aggregate gateway (AGW).

Figure 2 shows an IoT system reference architecture with IBC used for both AS and ISP security protection. From security point of view, both the AS and ISP may have their own security requirements for IoT services. Considering security credentials may be provided by either the AS or the ISP, there are three scenarios using IBC for IoT networks, as follows.

– using IBC in AS security protection scenario:

  For this scenario, security credentials for network access stored in IoT devices are provided and managed by the AS. IoT devices are authenticated by the AS when connecting to it. For example, an IoT device computes the IBS signature based on the private key provided by the AS and sends the signature to the AS. Accordingly, the AS can authenticate the IoT device based on the IBS signature provided in the authentication messages. If the verification is success, the AS sends the data from the IoT device to the IoT server;

– using IBC for ISP security protection:

  The IBC credentials stored in IoT devices are provided and managed by the ISP for service access. IoT devices are authenticated by the ISP based on the signature generated with IBC credentials;

– using IBC in both AS and ISP security protection scenario:

  The IBC security credentials stored in IoT devices are provided and managed by either the AS or by the ISP, or both jointly. The IoT device can be authenticated by both the AS and ISP with the same set of credentials.

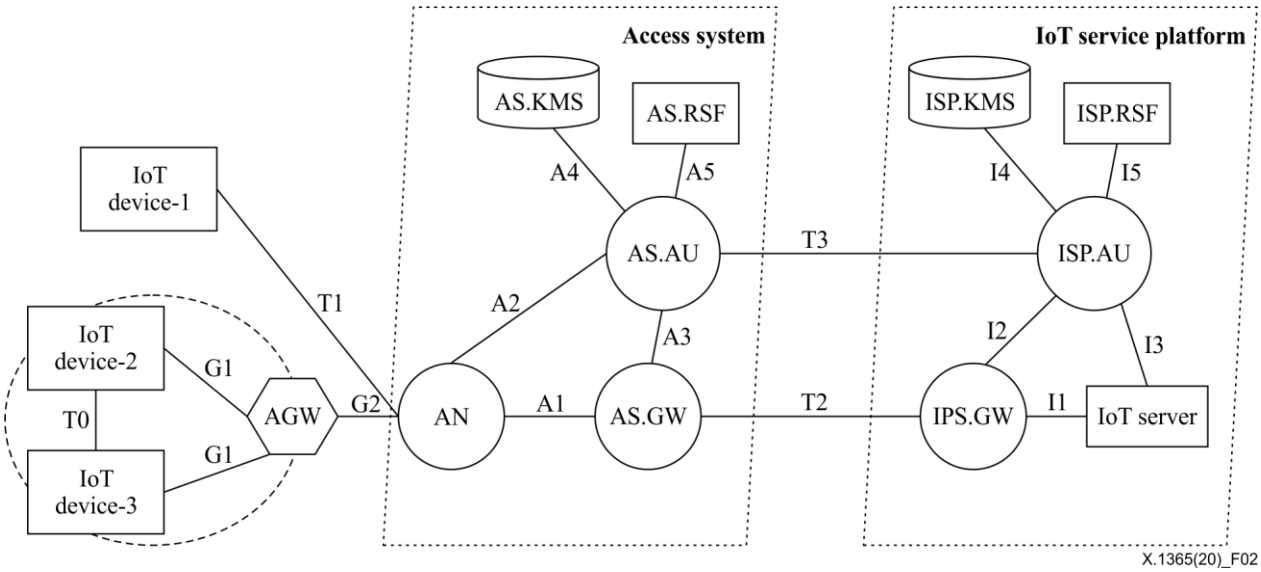

Figure 2 – IoT system architecture based on identity-based cryptography in both access system and IoT service platform security protection scenarios

The three scenarios described above cover most use cases with IBC for network and service access. However, there could be other scenarios that are outside the scope of this Recommendation.

IoT system architecture based on IBC consists of the following network functions (NF) and devices:

– access system (AS): a system of access for IoT devices or AGW, including access node (AS.AN), key management system (AS.KMS) function, authentication unit (AS.AU), revocation server function (AS.RSF) and gateway (AS.GW);

– IoT service platform (ISP): a platform for IoT service management, including key management system function (ISP.KMS), authentication unit (isp.au), revocation server function (ISP.RSF), gateway (ISP.GW) and IoT server. The ISP shall support key management, distribution, identity authentication, encryption or decryption and signature signing or verification, etc.;

– aggregate gateway (AGW): an aggregation node, responsible for an IoT device connection, aggregating and sending all IoT device data to the AS. The AGW plays the proxy role of data transmission between IoT devices and the AN;

– access node (AN): access node for IoT devices or AGW, can be a wireless or fixed network access point;

– key management system (KMS) function: a management system, responsible for key generation, distribution and update of the IBC keys and parameters for IoT devices and network functions;

– authentication unit (AU): an AU authenticates an IoT device based on IBC system;

– revocation server function (RSF): a server maintains an identity revocation list (IRL). Public keys or identities in the revocation list are excluded from usage;

NOTE – Both the AS and ISP may have their own KMS, AU and RSF.

– access system gateway (AS.GW): a network element connected to an IoT GW, responsible for IoT user data transmission;

– IoT gateway (IoT GW): a GW responsible for forwarding or aggregating data and transmitting data to an IoT server, or for forwarding data/signalling from an IoT server to IoT devices.

– IoT server: a server located on the IoT service provider side, collecting IoT data from the IoT GW;

– IoT device: an end device, used for data collection and connection establishment with an AN and IoT server, providing data protection, including key negotiation, encryption or decryption and signature signing or verification.

Functions of the reference points shown in Figure 2 are described as follows:

– G1: reference point between an IoT device and AGW, used for authentication and security communication;

– G2: reference point between an AGW and AN, used for signalling and data communication between the AGW and AN;

– T0: reference point between IoT devices, used for signalling and data exchange;

– T1: reference point between IoT devices and an AN, used for authentication and security communication;

– T2: reference points between AS.GW and ISP.GW, providing a user plane data tunnel between the AS.GW and ISP;

– T3: reference points between AS.AU and ISP.AU, for signalling exchange, including identity exchange or key provision;

– A1: reference points between AN and AS.GW, for user plane data tunnelling;

– A2: reference points between AS.AU and AN, for control plane signalling;

– A3: reference points between AS.AU and AS.GW, for GW allocation and management protocol in the AS;

– A4: reference points between AS.AU and AS.KMS, for the key provision protocol in the AS;

– A5: reference points between AS.AU and AS.RSF, for the identity or key revocation protocol in the AS;

– I1: reference points between IoT server and ISP.GW, for user plane data tunnelling;

– I2: reference points between ISP.AU and ISP.GW, for GW allocation and management protocol in the ISP;

– I3: reference points between ISP.AU and IoT server, for information exchange, such as service-related subscription information transferred from IoT server to ISP.AU, authentication notification message from ISP.AU to the IoT server;

– I4: reference points between ISP.AU and ISP.KMS, for the key provision protocol in the ISP;

– I5: reference points between ISP.AU and ISP.RSF, for the identity or key revocation protocol in the ISP.

8.2 Key management architecture

This clause describes the functional architecture required to support key management when using IBC mechanisms in IoT. Based on whether an IoT device has an embedded universal integrated circuit card (eUICC) [b-GSMA SGP.02] component, two types of key management architecture are considered: 1) IBC in IoT devices with eUICC; and 2) IBC in non-eUICC IoT devices.

If using IBC in IoT devices with eUICC, the architecture follows that of general eUICC remote provision defined in [b-GSMA SGP.02] by adding two new function entities, i.e., KMS and PPS. Depending on the location of the KMS, this case is further divided into the following two subcases:

1) KMS is managed by the entity that is also in charge of a mobile network operator (MNO) – see Figure 3;

2) KMS is managed by the entity that is in charge of subscription manager data preparation (SM-DP) – see Figure 4.

In both subcases, the keys including the private key and public parameters are generated when an MNO places a profile order. The keys are then provided remotely to eUICC devices as those keys installed following current remote key provision specification in [b-GSMA SGP.02]. Details of roles, associated functions and interfaces of eUICC remote provision can be found in [b-GSMA SGP.02]. Specification of profile, storage format and usage of these keys in eUICC is outside the scope of this Recommendation.

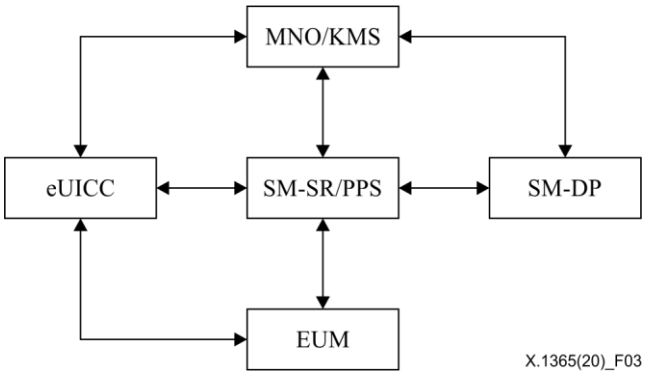

Figure 3 – Identity-based cryptography key management architecture A for IoT devices with embedded universal integrated circuit card

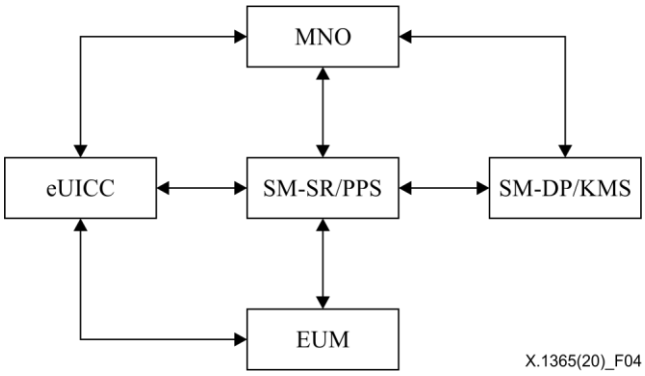

Figure 4 – Identity-based cryptography key management architecture B for IoT devices with embedded universal integrated circuit card

In the case of using IBC in non-eUICC IoT devices, a general architecture is depicted in Figure 5. The building blocks include the following:

– SecM: A security module (SecM) represents an element that can securely store keys and execute security mechanisms with the stored keys to complete security operations. An IoT device should have a SecM;

– IdP: An identity provider (IdP) is an entity that creates, maintains, and manages identity information;

– AuC: An authentication centre (AuC) offers entity authentication as a service.

The IdP relies on the authentication service provided by the AuC to authenticate IoT devices. After the initial authentication process, the IdP provides identity provision service to the SecM including identity creation, designation, replacement and revocation. After a new identity is created and designated to an IoT device, the IdP invokes the private key generation service provided by the KMS to generate the private key corresponding to the newly allocated ID and securely distribute the keys to the SecM. The IdP also extracts public parameters from the KMS and feed them into the PPS which publishes the public parameters to external entities. The IdP may also provide authentication service

to other entities by executing with the SecM specific authentication protocols including those defined in this Recommendation.

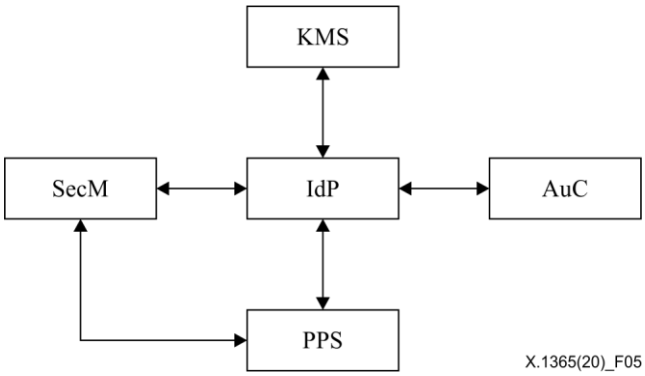

Figure 5 – Identity-based cryptography key management architecture for non-embedded universal integrated circuit card IoT devices

8.3 Identity naming

When using IBC technology for IoT services over a telecommunication network, identity naming can provide useful information to help operators to manage the network. Various pieces of information, e.g., service type, location, device ID and valid time can be embedded in an identity. Part of the information is necessary when using IBC technology, e.g., the valid time. With the identity information, the operator can optimize network management, e.g. by allocating a connection to specific network slices based on its service type. It is also easy to locate a device based on its location information. An example definition of the identity is presented in Appendix I.

8.4 Key management

Apart from the identity value, an IBC system involves three types of cryptographic key values: the MSK, the public parameters and the private key. The abstract syntax notation one (ASN.1) definition of these key structures is given in Annex B.

To manage these keys, the IBC system makes use of five key management operations:

1) system initialization operation;
2) device initialization;
3) public parameter lookup;
4) identity and key provisioning;
5) identity and key revocation.

The key management interoperability protocol (KMIP) [b-OASIS KMIP] can be utilized to exchange messages between the management entity and the KMS. However, extension necessary to KMIP to satisfy new requirements of the **IBSetup** and **IBExtract** function requires definition. For IoT devices with eUICC, standard procedures for remote key provision are used [b-GSMA SGP.02]. For non-eUICC IoT devices, the protocols for interactions between the SecMs and the management entities are defined based on hypertext transfer protocol (HTTP). Specifications of these operations are defined in Annex C.

The system initialization operation initializes an IBC system by generating the MSK and public parameters. A management entity, such as IdP, SM-DP or MNO, is assumed to be in charge of the IBC system initialization process. It establishes a secure channel with a KMS entity that implements the **IBSetup** function. The two parties execute the KMIP with the create key pair operation. The management entity provides necessary information for the KMS to invoke the **IBCSetup** function and generate the MSK and public parameters. The KMIP is extended to support setup functions

including those of various standardized IBC algorithms. Details of this operation are specified in clause C.1.

The device initialization operation is to prepare an IoT device for identity and key provision. There are two cases: initialization of IoT devices with eUICC and initialization of non-eUICC IoT devices. For devices with eUICC, it is required that eUICC completes the registration at subscription manager secure routing (SM-SR) and so is ready for profile download [b-GSMA SGP.02]. There is no extra operation required for standard eUICC devices. For non-eUICC IoT devices, the SecM should first register with AuC to get a provision ID (PROV.ID) and provision credential (PROV.CRED). This pair, PROV.ID/PROV.CRED, is used for entity authentication in the identity/key provision process. If IoT devices cannot establish a secure channel with IdP using transport layer security (TLS), it is further required that a key identity IdP.ID and a related public key IdP.PUK belonging to IdP or the public parameter should be installed in SecM during the device initialization process. Details of the operation are specified in clause C.2.

The public parameter lookup operation is to retrieve the IBC public parameters. An IoT device shall use the identity and key provision procedure to get the public parameters of the IBC system that it belongs to. It may follow the specification defined in clause 4 of [IETF RFC 5408] to retrieve the public parameters of another IBC system from the known PPS. Details of the operation are specified in clause C.3.

The identity and key provisioning operation includes the identity assignment, the private key extraction and the key distribution procedure. IoT devices after the initialization process only have a provisional identity. The IdP or SM-DP or MNO shall determine which identity is to be assigned to the requesting device and then communicate with the KMS to generate the corresponding private key and finally distribute the identity, the private key and the public parameters to the device securely. Details of the operation are specified in clause C.4.

The identity and key revocation operation is used when a stringent security policy requires that an identity should be revoked in a timely fashion. If an identity is revoked, the identity shall be set in the revoked status. If an entry queries the status of a revoked identity, IdP or SM-DP or MNO shall return the correct value as defined in the online identity status protocol (OISP). To check a batch of identity statuses more efficiently, an entity can retrieve the IRL from IdP or SM-DP or MNO regularly and store it locally, and the entity can check with the fresh IRL to determine whether an identity is revoked without querying status online for each identity. Details of the operation are specified in clause C.5.

8.5 Authentication

Authentication is the process of determining whether an entity (a device or a user) has the right to access certain resources. In telecommunication networks, there are two types of authentication related to IoT devices: network access authentication and service authentication. Network access authentication addresses whether a device is allowed to access the network, while service authentication addresses whether a device can access an ISP or not.

Authentication protocols built based on IBC technologies are suitable for IoT authentications in telecommunication networks. This is due to the fact that IBC can greatly reduce the burden on identity and key management for a massive number of IoT devices. Another advantage of IBC is that it enables distributed authentication, which not only significantly reduces authentication time, but also enables new application scenarios, e.g., device to device authentication, vehicle to vehicle authentication. For current telecommunications networks, such as 4G LTE networks, IBC can be used in authentication between IoT devices and ISPs. For 5G cellular networks, IBC can be used for both network access authentication and service access authentication. Current specification for 5G security, [b-ETSI TS 133.501], specifies a unified authentication framework that supports extensible authentication protocol (EAP) methods. Annex of [b-ETSI TS 133.501] further specifies how to use EAP-TLS in 5G for IoT networks.

The EAP framework is open and supports many authentication protocols, including EAP-TLS. Both symmetric and asymmetric keys are supported by EAP authentication methods.

As a relatively new public key technology, IBC is not well supported in existing authentication protocols. Therefore, in Annex D, four existing protocols are amended to support IBC in authentication:

1) clause D.1: one-pass secret transport protocol [ISO/IEC 11770-3];
2) clause D.2: TLS with raw public key [IETF RFC 8446];
3) clause D.3: EAP-TLS [IETF RFC 5216];
4) clause D.4: EAP-PSK [IETF RFC 4764IETF RFC 4764].

9 Security requirements

This Recommendation only focuses on security requirements of using IBC in IoT. General security threats and requirements for IoT are specified in [b-ITU-T X.1361]. As a cryptosystem, paramount security concerns are the integrity and authenticity of public keys used and secrecy of long-term and ephemeral secret keys used. An IBC system involves the following components: the MSK, public parameters, IDs, private keys and ephemeral secrets used in cryptographic operations.

9.1 Security requirements on master secret key

All private keys are generated by the MSK. In particular, if the MSK is compromised, the adversary has the ability to recreate any entity's private key and therefore can decrypt all messages protected with the corresponding public key or impersonate any entity. Any illegal access to the MSK would compromise the security of the IBC system. Hence, the MSK shall be stored in a hardened environment such as a hardware security module (HSM). Any access to the key shall be authenticated with strong security mechanisms.

9.2 Security requirement on public parameters

The public key is computed from the public parameters and an ID with the **IBDerivate** operation. Hence using a false set of public parameters generated by an adversary to encrypt a message or verify a signature will lead to compromise of secrecy of the encrypted message or reach a false conclusion of the originator of a signature. Hence, the public parameters shall be transmitted through a secure channel or with a valid signature. An entity shall verify the peer entity of the secure channel or verify the validity of the signature with regard to a trusted public key before accepting the public parameters.

9.3 Security requirement on identifier

In IoT, each entity owns an ID. If the same ID is assigned to more than one entity and the corresponding private key is provisioned to each entity, this could result in leakage of sensitive information or impersonation attacks. Hence, each device shall be assigned a unique ID.

9.4 Security requirement for private key

The private key could be leaked if the security environment of an IoT device is compromised. Hence, the private key shall be distributed through a secure channel and stored in a secure environment.

9.5 Security requirement for ephemeral secrets

Ephemeral secrets, such as the random secret used in encryption or signature processes, could be leaked if the security environment of an IoT device is compromised. Hence, randomness of ephemeral secrets shall be guaranteed.

Annex A

Generic formulation and algorithms of identity-based cryptography

(This annex forms an integral part of this Recommendation.)

This annex gives a generic formulation of IBC and provides a list of IBC algorithms that are supported in this Recommendation. The algorithms that follow this generic formulation but that are not listed below may also be easily included in the future as extensions to this framework. The generic formulation specified here also guides the descriptions of related key data structures, key management operations, as well as authentication and key establishment protocols defined in Annex B through Annex D.

An IBC cryptosystem involves the following types of key data, where the categorization of these keys follows [ISO/IEC 18033-5]:

- *ib.msk*: the MSK is the secret value used by the KMS to compute an identity-based private key. *ib.msk* is generated during the system initialization process and is known by KMS only;
- *ib.mpk*: the MPK that is uniquely determined by the corresponding MSK. *ib.mpk* is computed by KMS during the system initialization process;
- *ib.sysparam*: the system parameters for cryptographic computation including a selection of a particular cryptographic scheme or function from a family of cryptographic schemes or functions, or from a family of mathematical spaces. *ib.sysparam* is chosen by KMS during the system initialization process;
- *ib.pubparam*: the public parameters are the combination of the system parameters *ib.sysparam* with the MPK *ib.mpk*. This type of key is defined to provide a unified view among International Standards such as [ISO/IEC 18033-5] and RFCs related to IBC such as [IETF RFC 5091];
- *ib.prk*: the identity-based private key, which is generated by KMS with *ib.msk* and *ib.pubparam*, corresponding to an identifier *ID*;
- *ib.pub*: the identity-based public key, which is computed from an identifier *ID* and *ib.pubparam* through a function defined by an identity-based cryptographic scheme.

An IBC cryptosystem may include the following functions, which are specified with inputs and outputs:

IBSetup

input: security parameter

output: *ib.pubparam, ib.msk*

IBExtract

input: *ib.pubparam, ib.msk, ID*

output: *ib.prk*

IBDerivate

input: *ib.pubparam, ID*

output: *ib.puk*

IBEnc

input: *ib.pubparam, ID,* message *M*

output: ciphertext *C*

IBDec

input: *ib.pubparam*, *ID*, *ib.prk*, ciphertext *C*

output: plantext *M* or error

IBSign

input: *ib.pubparam*, *ID*, *ib.prk*, message *M*

output: signature *S*

IBVerify

input: *ib.pubparam*, *ID*, message *M*, signature *S*

output: valid or invalid

This Recommendation shall support the use of the following identity-based algorithms:

– BB1-KEM (key encapsulation mechanism (KEM)) [IETF RFC 5091];
– BF-IBE [IETF RFC 5091];
– SK-KEM [IETF RFC 6508];
– SM9-IBE [b-GM/T 0044.2];
– Cha-Cheon-IBS (IBS2) [ISO/IEC 14888-3];
– ECCSI (elliptic curve-based certificateless signatures for identity-based encryption) [IETF RFC 6507];
– Hess-IBS (IBS1) [ISO/IEC 14888-3];
– SM9-IBS (Chinese IBS) [ISO/IEC 14888-3];
– Fujioka-Suzuki-Ustaoglu-AKA (authenticated key agreement (AKA)) [ISO/IEC 11770-3];
– Smart-Chen-Cheng-AKA [ISO/IEC 11770-3];
– SM9-AKA [b-GM/T 0044.2];
– Wang-AKA [b-IEEE P1363.3].

All of these algorithms are based on the discrete logarithm assumption and typically are implemented on the point group over an elliptic curve. Many of these algorithms also make use of cryptographic pairing over an elliptic curve [b-Galbraith]. A cryptographic pairing e is an efficiently computable bilinear map e: $G1 \times G2 \rightarrow G3$, satisfying the equation:

$$e([a]\mathrm{P1}, [b]\mathrm{P2}) = \mathrm{e}(\mathrm{P1}, \mathrm{P2})^{a*b}$$

where P1 and P2 are the generator of cyclic group $G1$ and $G2$, respectively. $[a]$P1 denotes a times group operations with P1; similarly $[b]$P2 is the group operation with P2.

A cryptographic pairing can be instantiated by the Weil pairing, Tate pairing, optimal Ate pairing, etc., over pairing-friendly elliptic curves [b-Freeman]. The commonly used pairing-friendly elliptic curves include supersingular elliptic curves, Barreto-Naehrig (BN) curves, Barreto-Lynn-Scott embedding degree 12 (BLS-12) curves, Kachisa-Schaefer-Scott embedding degree 16 (KSS-16) curves, Kachisa-Schaefer-Scott embedding degree 18 (KSS-18) curves and Barreto-Lynn-Scott embedding degree 24 (BLS-24) curves [b-Freeman]. All these curves E are based on a prime field, a finite field of prime characteristic p, F_p, where p is a prime integer. $G1$ is the point subgroup on curve E. $G2$ is either the same as $G1$ if the supersingular curves are used or is a point subgroup on the twist curve E. E is constructed from some extension field of the based field F_p. $G3$ is the field extension F_p^k of the base field F_p, where k is the embedding degree.

There IBC algorithms are constructed with other mathematical mechanisms such as lattices, e.g., [b-Ducas]. This type of algorithm is efficient regarding computation, while having larger key and output size than those based on the discrete logarithm over elliptic curves. The algorithms are generally believed to be resistant to attacks running on quantum computers. However, algorithms of this category are still under development. Hence it seems premature to consider them for standardization, but those lattice-based IBC algorithms may be considered for future inclusion.

Annex B

Identity-based cryptography key data specification

(This annex forms an integral part of this Recommendation.)

Using the standard ASN.1 method, [IETF RFC5408] has defined a generic structure for system parameters, including *ib.pubparam* and other auxiliary information, and [IETF RFC 5091] has defined two sets of key data structures including *ib.msk* and *ib.prk* for two IBE algorithms, i.e., BF-IBE and BB1-IBE. While maintaining compatibility with the existing definitions, this Recommendation extends the system parameter definition and defines new key data structures to support more algorithms and various efficient implementations with different curves and pairings.

A generic system parameter structure is defined as follows:

```
IBSysParams ::= SEQUENCE {
        version                 INTEGER { v3(3) },
        domainName              IA5String,
        domainSerial            INTEGER,
        validity                ValidityPeriod,
        ibPublicParameters      IBPublicParameters,
        ibIdentityType          OBJECT IDENTIFIER,
        ibParamExtensions       [0] IMPLICIT IBParamExtensions OPTIONAL,
        signatureAlgorithm      [1] IMPLICIT AlgorithmIdentifier OPTIONAL,
        signature               [2] IMPLICIT BIT STRING OPTIONAL
}
```

IBSysParams corresponds to the IBESysParams definition in [IETF RFC 5408], but the version is changed to v3 (3) and two extra fields are added. districtName and districtSerial have been renamed as domainName and domainSerial, respectively. The definition of IBPublicParameter has been modified from the OCTET STRING type to the newly defined IBParameterData type, which is a CHOICE determined by the value of pkgAlgorithm. This definition removes unnecessary double-encoding caused by the previous definition, namely, encoding publicParameterData as a SEQUENCE of, for example, BFPublicParameters and further encoding the result as an OCTET STRING. Except the two new fields, the meaning of other fields remains unchanged as in [IETF RFC 5408]. The meanings of the two new fields are as follows:

- signature algorithm designates the signature algorithm used to generate the signature value. This field is optional, as the signature field is not mandatory;
- signature field contains the digital signature computed upon the ASN.1 distinguished encoding rules (DER) result from field version to *ibParamExtensions*. This field is encoded as BIT STRING and is optional.

If present, the signature field is used to help an entity to check the authenticity of the system public parameters without resorting other methods. For example, if an IoT device does not have the capability to establish a TLS-based secure channel as required in [IETF RFC 5408] to retrieve the public parameters of another IBC system, it may query its PPS with HTTP. In this case, the serving PPS shall sign the requested public parameters with its private signing key. The IoT device can verify the signature to check the authenticity of the response. If a PPS is publishing the public parameters of another IBC system to its serving entities, it is recommended that the signing message be treated

as an identity and the **IBExtract** algorithm used as the signature algorithm to generate the private key as the corresponding signature value. In this way, the IoT devices verify whether the signature value is a valid private key corresponding to the ASN.1 DER result from field version to *ibParamExtensions* and do not need an extra verification public key to verify the signature.

```
ValidityPeriod ::= SEQUENCE {
        notBefore       GeneralizedTime,
        notAfter        GeneralizedTime
}
IBPublicParameters ::= SEQUENCE SIZE (1..MAX) OF IBPublicParameter
IBPublicParameter ::= SEQUENCE {
        pkgAlgorithm            OBJECT IDENTIFIER,
        publicParameterData     IBParameterData
}
```

The value of publicParameterData is defined by pkgAlgorithm. It can be one of following choices.

```
IBParameterData ::= CHOICE {
        bb1ParameterData    [0] IMPLICIT BB1PublicParameters,
        bfParameterData     [1] IMPLICIT BFPublicParameters,
        eccsiParameterData  [2] IMPLICIT ECCSIPublicParameters,
        skParameterData     [3] IMPLICIT SKPublicParameters,
        sm9ParameterData    [4] IMPLICIT SM9PublicParameters
}
IBParamExtensions ::= SEQUENCE OF IBParamExtension
IBParamExtension ::= SEQUENCE {
        ibParamExtensionOID     OBJECT IDENTIFIER,
        ibParamExtensionValue   OCTET STRING
}
AlgorithmIdentifier ::= SEQUENCE {
        algorithm       OBJECT IDENTIFIER,
        parameters      ANY DEFINED BY algorithm OPTIONAL
}
```

In [IETF RFC 5091], two sets of MSK, public parameters and private key block, i.e.:

- BB1MasterSecret, BB1PublicParameters, BB1PrivateKeyBlock;
- BFMasterSecret, BFPublicParameters, BFPrivateKeyBlock are defined for BF and BB1 key generation function. These designations only fit with implementations of the functions with symmetric pairings on supersingular elliptic curves defined over prime fields. This Recommendation specifies new structures with version changed to v3 to support implementations of these algorithms with asymmetric pairings. For symmetric pairings over supersingular elliptic curves, the corresponding field in BB1 and BF key data structures remains unchanged as in [IETF RFC 5091]. Three more sets of key data structures are defined for ECCSI, SM9, and SK-KEM. Respectively;

```
BB1MasterSecret ::= SEQUENCE {
        version     INTEGER { v3(3) },
        alpha       INTEGER,
        beta        INTEGER,
        gamma       INTEGER
}
```

– for implementations with asymmetric pairings, alpha shall be s1, beta shall be s2 and gamma shall be s3 in clause 9.3 of [ISO IEC 18033-5];

```
BB1PublicParameters ::= SEQUENCE {
        version     INTEGER { v3(3) },
        curve       OBJECT IDENTIFIER,
        hashfcn     OBJECT IDENTIFIER,
        pairing     PAIRING OPTIONAL,
        p           INTEGER OPTIONAL,
        q           [0] IMPLICIT INTEGER OPTIONAL,
        pointP      FpPoint,
        pointQ      [1] EXPLICIT FpxPoint OPTIONAL
        pointP1     FpPoint,
        pointP2     [2] EXPLICIT FpxPoint OPTIONAL,
        pointP3     FpPoint,
        v           FpxElement
}
```

– pairing specifies which type of bilinear map shall be used with generated parameters. Three types of pairing are supported: Weil pairing; Tate pairing; and optimal Ate pairing.

– p and q become optional. For some types of curves, such as BN, BLS-12, etc., p and q are pre-determined by curve object identifiers (OIDs) and hence it is unnecessary to specify them again.

– pointP andpointQ, for implementation with asymmetric pairings, shall be Q1 in *G*1 and Q2 in *G*2 in clause 9.3 of [ISO IEC 18033-5]. For symmetric pairings, pointP equals to pointQ, so pointQ is OPTIONAL.

– pointP1 and pointP3, for implementation with asymmetric pairings, shall be R and T in clause 9.3 of [ISO/IEC 18033-5].

– pointP2, for implementation with asymmetric pairings such as the optimal Ate pairing over BN curves, takes value from an extension field of F_p. pointP2 is optional because if v is given, pointP2 is unnecessary for the BB1-KEM algorithm to carry through.

– v is the pairing result, which is an element from the extension field of F_p. For implementation with asymmetric pairings, such as the optimal Ate pairing over BN curves, the extension field is F_p^k, where k is the embedding degree. In this case, v shall be J in clause 9.3 of [ISO/IEC 18033-5].

– the meaning of other fields remains unchanged as in [IETF RFC 5091].

```
PAIRING ::= ENUMERATED {
        weil        (1),    --Weil pairing
        tate        (2),    --Tate pairing
        optimalAte  (3)     --Optimal Ate pairing
}
FpPoint ::= SEQUENCE {
        x   INTEGER,
        y   INTEGER
}
```

FpPoint defines a point on an elliptic curve over a prime field. A point has two coordinates, which are designated as the x-coordinate and y-coordinate. Both coordinates take big integer values.

```
FpxPoint ::= CHOICE {
        fpPoint     [1] EXPLICIT FpPoint,
        fp2Point    [2] EXPLICIT Fp2Point,
        fp3Point    [3] EXPLICIT Fp3Point,
        fp4Point    [4] EXPLICIT Fp4Point
}
```

– Fp2Point defines a point on an elliptic curve over a field F_p^2. Each coordinate of a point takes value from an element of F_p^2.

– Fp3Point defines a point on an elliptic curve over a field F_p^3. Each coordinate of a point takes value from an element of F_p^3.

– Fp4Point defines a point on an elliptic curve over a field F_p^4. Each coordinate of a point takes value from an element of F_p^4.

```
Fp2Point ::= SEQUENCE {
        x   Fp2Element,
        y   Fp2Element
}
```

– Fp2Point defines a point on an elliptic curve over a field F_p^2. A point has two coordinates which are named as x-coordinate and y-coordinate. Both coordinates take values from F_p^2.

```
Fp3Point ::= SEQUENCE {
        x   Fp3Element,
        y   Fp3Element
}
```

– Fp3Point defines a point on an elliptic curve over a field F_p^3. Both coordinates of a point take values from F_p^3.

```
Fp4Point ::= SEQUENCE {
        x   Fp4Element,
        y   Fp4Element
}
```

– Fp4Point defines a point on an elliptic curve over a field F_p^4. Both coordinates of a point take values from F_p^4.

```
Fp2Element ::= SEQUENCE {
    a    INTEGER,
    b    INTEGER
}
```

– Fp2Element defines an element from a field F_p^2, which is represented as a+bα where α is non-quadratic root in F_p.

```
Fp3Element ::= SEQUENCE {
    a    INTEGER,
    b    INTEGER,
    c    INTEGER
}
```

– Fp3Element defines an element from a field F_p^3, which is represented as a+bβ+cβ^2 where β is non-cubic root in F_p.

```
Fp4Element ::= SEQUENCE {
    a    Fp2Element,
    b    Fp2Element
}
```

– Fp4Element defines an element from a field F_p^4, which is represented as tower of two elements from F_p^2.

```
FpxElement ::= CHOICE {
    fp2Elemt    [1] EXPLICIT Fp2Element,
                --for super singular elliptic curve implementation
    fp12Elemt   [2] EXPLICIT Fp12Element,
                --using F_p->F_p^2->F_p^6->F_p^12 tower representation
    fp16Elemt   [3] EXPLICIT Fp16Element,
                --using F_p->F_p^2->F_p^4->F_p^8->F_p^16 tower representation
    fp18Elemt   [4] EXPLICIT Fp18Element,
                --using F_p->F_p^3->F_p^6->F_p^18 tower representation
    fp24Elemt   [5] EXPLICIT Fp24Element
                --using F_p->F_p^2->F_p^6->F_p^12->F_p^24 tower representation
}
```

– FpxElement defines the tower representation of an element in $G3$. The pairing e maps two inputs from $G1$ and $G2$, respectively, to an element in $G3$. For the commonly used pairing-friendly curves, elements in $G3$ are normally represented in a tower method. For different embedding degrees, there could be different tower representations. This Recommendation defines a commonly used tower representation of elements in field with embedding degree 12, 16, 18 and 24.

Fp12Element ::= SEQUENCE {

a Fp6Element,

b Fp6Element

}

– Fp12Element defines an element of F_p^12 with a 2x3x2 tower representation and shall be used in implementation with BN curves or BLS-12 curves or BLS-24 curves.

Fp6Element ::= SEQUENCE {

a Fp2Element,

b Fp2Element,

c Fp2Element

}

– Fp6Element defines an element of F_p^6 with a 3x2 tower representation and shall be used in implementation with BN curves or BLS-12 curves or BLS-24 curves.

Fp16Element ::= SEQUENCE {

a Fp8Element,

b Fp8Element

}

– Fp16Element defines an element of F_p^16 with a 2x2x2x2 tower representation and shall be used in implementation with KSS-16 curves.

Fp8Element ::= SEQUENCE {

a Fp4Element,

b Fp4Element

}

– Fp8Element defines an element of F_p^8 with a 2x2x2 tower representation and shall be used in implementation with KSS-16 curves.

Fp18Element ::= SEQUENCE {

a Fp6bElement,

b Fp6bElement,

c Fp6bElement

}

– Fp18Element defines an element of F_p^18 with a 3x2x3 tower representation and shall be used in implementation with KSS-18 curves.

Fp6bElement ::= SEQUENCE {

a Fp3Element,

b Fp3Element

}

– Fp6bElement defines an element of F_p^6 with a 2x3 tower representation and shall be used in implementation with KSS-18 curves.

Fp24Element ::= SEQUENCE {

a Fp12Element,

b Fp12Element

}

– Fp24Element defines an element of F_p^24 with a 2x2x3x2 tower representation and shall be used in implementation with BLS-24 curves.

BB1PrivateKeyBlock ::= SEQUENCE {

version INTEGER { v3(3) },

pointD0FpxPoint,

pointD1FpxPoint

}

– The meanings of pointD0 and pointD1 remain unchanged as in [IETF RFC 5091], but they are taken from G2 if BB1-KEM is implemented with asymmetric pairings. In this case, pointD0 and pointD1 shall be dID0 and dID1 respectively in clause 9.3 of [ISO/IEC 18033-5].

BFMasterSecret ::= SEQUENCE {

version INTEGER {v3(3) },

masterSecret INTEGER

}

– The meaning of each field remains unchanged as in [IETF RFC 5091].

BFPublicParameters ::= SEQUENCE {

version INTEGER { v3(3) },

curve OBJECT IDENTIFIER,

hashfcn OBJECT IDENTIFIER,

pairing PAIRING OPTIONAL,

p INTEGER OPTIONAL,

q [0] IMPLICIT INTEGER OPTIONAL,

pointP FpxPoint,

pointPpub FpxPoint

}

– The meaning of each field remains unchanged as in [IETF RFC 5091], but pointP and pointPpub is taken from *G2* if BF-IBE is implemented with asymmetric pairings. In this case, pointP and pointPpub shall be Q and R respectively in clause 8.2 of [ISO/IEC 18033-5].

BFPrivateKeyBlock ::= SEQUENCE {

version INTEGER { v3(3) },

privateKey FpPoint

}

– The meanings of each field remain unchanged as in [IETF RFC 5091]. For implementations with asymmetric pairings, privateKey shall be skID in clause 8.2 of [ISO/IEC 18033-5].

```
ECCSIMasterSecret ::= SEQUENCE {
        version      INTEGER {v3(3) },
        masterSecret  INTEGER
}
```

– masterSecret shall be KSAK in [IETF RFC 6507].

```
ECCSIPublicParameters ::= SEQUENCE {
        version      INTEGER { v2(2) },
        curve        OBJECT IDENTIFIER,
        hashfcn      OBJECT IDENTIFIER,
        pointP       FpPoint,
        pointPpub    FpPoint
}
```

– pointP shall be G in [IETF RFC 6507].

– pointPpub shall be the KMS public authentication key (KPAK) in [IETF RFC 6507].

```
ECCSIPrivateKeyBlock ::= SEQUENCE {
        version      INTEGER { v2(2) },
        ssk          INTEGER ,
        pvt          OCTET STRING
}
```

– ssk and pvt shall be a secret signing key (SSK) and public verification token (PVT) in [IETF RFC 6507], respectively.

```
SKMasterSecret ::= SEQUENCE {
        version      INTEGER {v3(3) },
        masterSecret  INTEGER
}
```

– masterSecret shall be z_T in [IETF RFC 6508] and s in clause 9.2 of [ISO/IEC 18033-5].

```
SKPublicParameters ::= SEQUENCE {
        version          INTEGER { v3(3) },
        curve            OBJECT IDENTIFIER,
        hashfcn          OBJECT IDENTIFIER,
        pairing          PAIRING OPTIONAL,
        p                INTEGER OPTIONAL,
        q                [0] IMPLICIT INTEGER OPTIONAL,
        pointP1          FpPoint,
        pointP1pub       [1] EXPLICIT FpPoint OPTIONAL,
        pointP2          [2] EXPLICIT FpxPoint OPTIONAL,
        pointP2pub       [3] EXPLICIT FpxPoint OPTIONAL,
```

```
        v               [4] EXPLICIT FpxElement
}
```

– For implementations with symmetric pairings over supersingular curves, p and q are defined in [IETF RFC 5091]. For implementations with asymmetric pairings, p and q are pre-determined by the curve used and become optional.

– pointP1 shall be P in [IETF RFC 6508] and Q1 in *G*1 in clause 9.2 of [ISO/IEC 18033-5].

– pointP1pub shall be Z_T in [IETF RFC 6508] and R in clause 9.2 of [ISO/IEC 18033-5]. pointP1pub may be unnecessary for other algorithms, such as signature algorithms, based on the Sakai-Kasahara (SK) generation function, so it is optional.

– pointP2 shall be Q2 in *G*2 in clause 9.2 of [ISO/IEC 18033-5] if the SK-KEM is implemented with asymmetric pairings. pointP2 is not necessary for SK-KEM to carry through, so it is optional.

– pointP2pub shall be [*ib.msk*]Q2, which is unnecessary for SK-KEM, but may be necessary for other algorithms, such as signature algorithms, based on the SK key generation function, so it is optional.

```
SKPrivateKeyBlock ::= SEQUENCE {
        version     INTEGER { v3(3) },
        privateKey  FpxPoint
}
```

– privateKey shall be RSK in [IETF RFC 6508] and skID in clause 9.2 of [ISO/IEC 18033-5].

```
SM9MasterSecret ::= SEQUENCE {
        version         INTEGER {v3(3) },
        masterSecret    INTEGER
}
```

– masterSecret shall be *ib.msk* which is U defined in clause 7.4 of [b-ISO/IEC 14888-3a].

```
SM9PublicParameters ::= SEQUENCE {
        version         INTEGER { v3(3) },
        curve           OBJECT IDENTIFIER,
        hashfcn         OBJECT IDENTIFIER,
        pairing         PAIRING OPTIONAL,
        p               INTEGER OPTIONAL,
        q               [0] IMPLICIT INTEGER OPTIONAL,
        pointP1         FpPoint,
        pointP1pub      [1] EXPLICIT FpPoint OPTIONAL,
        pointP2         [2] EXPLICIT FpxPoint OPTIONAL,
        pointP2pub      [3] EXPLICIT FpxPoint OPTIONAL,
        v               [4] EXPLICIT FpxElement
}
```

– For implementations with symmetric pairing over supersingular curves, p and q are as defined in [IETF RFC 5091]. For implementations with asymmetric pairings, p and q are pre-determined by the curve used.
– pointP1 shall be P in clause 7.4 of [ISO/IEC 14888-3].
– pointP1pub is unnecessary for SM9-IBS, but necessary for SM9-IBE and in this case pointP2pub shall be [*ib.msk*]P.
– pointP2 shall be Q in clause 7.4 of [ISO/IEC 14888-3]. pointP2 is not necessary for **SM9-IBE** to carry through, so it is optional.
– pointP2pub shall be V in clause 7.4 of [ISO/IEC 14888-3a]. pointP2pub is not necessary for **SM9-IBE** to carry through, so it is optional.

```
SM9PrivateKeyBlock ::= SEQUENCE {
        version     INTEGER { v3(3) },
        privateKey  FpxPoint
}
```

– privateKey shall be X in in clause 7.4 of [ISO/IEC 14888-3a] for signature and shall be the *ib.prvk* in *G*1 for SM9-IBE and SM9-AKA.

The BFMasterSecret, BFPublicParameters, and BFPrivateKeyBlock definition shall be used for those algorithms using the Sakai-Ohgishi-Kasahara (SOK) key generation such as BF-IBE, Cha-Cheon-IBS, Hess-IBS, Fujioka-Suzuki-Ustaoglu-AKA, Smart-Chen-Cheng-AKA, and Wang-AKA. The BB1MasterSecret, BB1PublicParameters, and BB1PrivateKeyBlock definition shall be used for those algorithms using the BB1 key generation such as BB1-KEM. The SKMasterSecret, SKPublicParameters and SKPrivateKeyBlock shall be used for SK-KEM and possibly other algorithms based on the SK key generation function. The SM9MasterSecret, SM9PublicParameters and SM9PrivateKeyBlock shall be used for SM9 algorithms including SM9-IBE, SM9-IBS, and SM9-AKA. The ECCSIMasterSecret, ECCSIPublicParameters, and ECCSIPrivateKeyBlock shall be used for ECCSI.

If the private key needs to be protected, the EncryptedPrivateKeyInfo structure as defined in [IETF RFC 5958] shall be used.

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
        encryptionAlgorithm     EncryptionAlgorithmIdentifier,
        encryptedData           EncryptedData
}

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

EncryptedData ::= OCTET STRING

AlgorithmIdentifier ::= SEQUENCE {
        algorithm     OBJECT IDENTIFIER,
        parameters    ANY DEFINED BY algorithm OPTIONAL
}
```

Annex C

Key management operations

(This annex forms an integral part of this Recommendation.)

In an IBC system, key management operations include system initialization, identity or private key provision, identity or private key revocation and system parameter publication. System initialization involves a step to invoke the **IBSetup** function, and private key provision involves a step to invoke the **IBExtract** function. These operations require interactions between a management entity and the KMS. This Recommendation uses KMIP to exchange messages between these two parties. The necessary extension is specified to satisfy the new requirements of the supported **IBSetup** and **IBExtract** algorithms can be found in Appendix II. The protocols for interactions between the SecMs and the management entities are defined based on HTTP for non-eUICC IoT devices. For eUICC, [b-GSMA SGP.02] standards are used and extended if necessary.

C.1 System initialization

In each IBC system, a system initialization process should be completed before providing KMSs to its users. In the process, the KMS executes one or more **IBSetup** functions to generate one or more sets of *ib.msk* and *ib.pubparam* key pairs. The method to harden the security of the KMS lies outside the scope of this Recommendation. As a good practice, *ib.msk* shall be generated and stored in an HSM. If possible, a distributed key generation scheme that applies a secret sharing scheme to split *ib.msk* and spread the secret shares and the private key generation function across several KMSs may be deployed. In this case, only if more than a threshold number of KMSs is functioning appropriately, can a private key corresponding to an ID be generated correctly.

See Figure C.1.

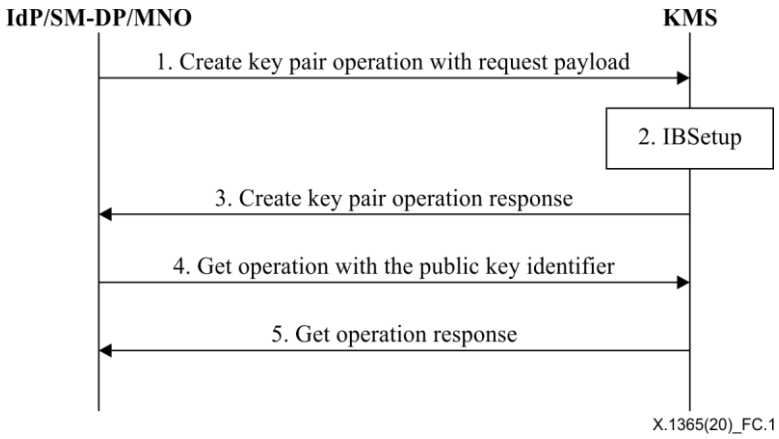

Figure C.1 – System initialization with key management interoperability protocol

Start conditions:

It is assumed that IdP/SM-DP/MNO plays the role of a system initiator and is in charge of the system initialization process. Before IdP/SM-DP/MNO can invoke the **IBSetup** function in a KMS, the following conditions shall be satisfied.

a) There is a secure channel established between IdP/SM-DP/MNO and KMS.

b) IdP/SM-DP/MNO has completed an authentication process with KMS, and the authenticated IdP/SM-DP/MNO is authorized to perform the **IBSetup** request.

Procedure:

1) IdP/SM-DP/MNO shall prepare the Request Payload and invoke the create key pair operation to send encoded request message to KMS;
2) KMS shall check the validity of the request and that IdP/SM-DP/MNO is authorized to invoke this operation. If any of these conditions is not satisfied, then KMS shall return a response indicating a failure. Otherwise, KMS shall execute **IBSetup** with parameters specified within the request;
3) KMS shall return to IdP/SM-DP/MNO the execution response. If the operation is a success, then KMS shall at least return a private key unique ID to *ib.msk* and a public key unique ID to *ib.pubparam*, respectively;
4) optionally, if the create key pair operation is successful, IdP/SM-DP/MNO may invoke the get operation with the public key unique ID obtained from the last response to retrieve the public parameters *ib.pubparam*;
5) KMS shall return the key value of the newly generated public parameters.

The extension of KMIP to support this operation can be found in Appendix II.

End condition: The KMS is initialized successfully and IdP/SM-DP/MNO has the private key unique ID and public key unique ID to access the generated MSK *ib.msk* and the public parameters *ib.pubparam*, respectively. IdP/SM-DP/MNO shall use the private key unique ID to call the sign operation to generate identity private keys and shall use the public key unique ID to call the get operation to retrieve the public parameters.

C.2 Device initialization

Device initialization operation is to prepare the device for identity and key provision. For eUICC and other non-eUICC IoT devices, different device initialization procedures are followed.

C.2.1 Case 1: Initialization for eUICC

For eUICC, the identity and corresponding private key *ib.prk* and public parameters *ib.sysparam* are downloaded in an issuer security domain (ISD) profile. Hence, after device initialization process, eUICC should be ready for ISD profile creation. Following [b-GSMA SGP.02], the registration operation shall be completed. The following is a reiteration of clause 3.5.1 of [b-GSMA SGP.02].

- eUICC Registration at SM-SR

Start condition:

a. eUICCs are produced and a provisioning profile is loaded and active in the provisioning operators network. They are tested and ready for shipment. Each eUICC has a corresponding eUICC information set (EIS).

Procedure:

1) the eUICC manufacturer (EUM) sends an eUICC registration request to the selected SM-SR, containing the EIS;
2) the SM-SR stores the EIS in its database, with eUICC-ID (EID) as the key parameter;
3) the SM-SR confirms the successful registration towards the EUM. The confirmation message includes the EID.

End condition: The eUICC is registered at the SM-SR and ready for Profile download. It can now be shipped to the machine to machine Device manufacturer.

C.2.2 Case 2: Initialization for non-eUICC IoT devices

For non-eUICC IoT devices, the following registration operation shall be completed.

- SecM Registration at AuC

Start condition:

a. The SecM is produced and the IoT device shall be able to communicate with IdP in the operators network.

Procedure:

1) The SecM sends a SecM provision data acquisition request to the AuC;
2) The AuC generates a provision ID (PROV.ID) and a related authentication credential (PROV.CRED) for the requesting SecM;
3) The AuC sends PROV.ID and PROV.CRED to the SecM. In the same message, the AuC also sends a key identity IdP.ID and a related public key IdP.PUK or *ib.sysparam* to the SecM if the SecM has no capability to carry out the TLS protocol;
4) The SecM securely stores PROV.ID and PROV.CRED, and store IdP.ID and IdP.PUK or *ib.sysparam* if provided. The SecM shall protect IdP.ID and IdP.PUK or *ib.sysparam* from authorized change.

End condition: The SecM is registered at the AuC and ready for identity and key provision.

C.3 Public parameter lookup

An entity shall use the identity or key provision procedure to get the public parameters for the IBC system with which the entity has registered. An entity, which can be an IoT device or a management entity of an IBC system, shall follow the specification defined in clause 4 of [IETF RFC 5408] to retrieve the public parameters of another IBC system from the known PPS. The IBESysParams in the response in [IETF RFC 5408] shall be replaced with IBSysParams defined in this Recommendation. [IETF RFC 5408] assumes that the querying IoT device can establish a TLS-based secure channel with the requested PPS. If such a requirement cannot be satisfied, the signatureAlgorithm and signature field in IBSysParams shall exist and be valid. Once the IBSysParams has been retrieved, a proper signature verification process shall be followed, and only if the signature in IBSysParams is valid and the signature verifying public key is authentic and valid, shall the retrieved public parameters be accepted.

C.4 Identity and key provisioning

Identity and key provision include the identity assignment, the private key extraction and the key distribution procedure. Devices after the initialization process only have a provisional identity. The IdP or SM-DP or MNO shall determine which identity is to be assigned to the requesting device and then communicate with the KMS to generate the corresponding private key and finally distribute the identity, the private key and the public parameters to the device securely.

See Figure C.2.

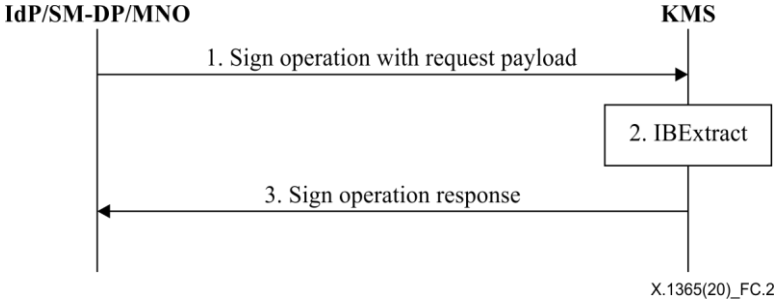

Figure C.2 – Private key generation with key management interoperability protocol

- **Private key generation with KMIP**

Start conditions:

Assuming that the IdP/SM-DP/MNO plays the role of generating private key *ib.prk*. Before IdP/SM-DP/MNO can invoke the IBExtract function in a KMS, the following conditions shall be satisfied.

a. There is a secure channel established between IdP/SM-DP/MNO and KMS.

b. The IdP/SM-DP/MNO has completed an authentication process to the KMS and the authenticated IdP/SM-DP/MNO is authorized to perform the IBExtract request.

Procedure:

1) the IdP/SM-DP/MNO shall prepare the request payload and invoke the sign operation to send the encoded request message to the KMS.

2) the KMS shall check the validity of the request and that the IdP/SM-DP/MNO is authorized to invoke this operation. If any of these conditions is not satisfied, then the KMS shall return a response indicating a failure. Otherwise, the KMS shall execute **IBExtract** with *ib.msk*, *ib.pubparam*, and parameters specified within the request.

3) the KMS shall return the execution response to the IdP/SM-DP/MNO. If the operation is a success, then the KMS shall return the generated private key *ib.prk* in the form of IBPrivateKeyBlock which is a CHOICE defined with ASN.1 as follows:

```
IBPrivateKeyBlock ::= CHOICE {
    bb1PrivateKeyBlock      BB1PrivateKeyBlock,
    bfPrivateKeyBlock       BFPrivateKeyBlock,
    eccsiPrivateKeyBlock    ECCSIPrivateKeyBlock,
    skPrivateKeyBlock       SKPrivateKeyBlock,
    sm9PrivateKeyBlock      SM9PrivateKeyBlock
}
```

The extension of KMIP to support this operation can be found in Appendix II.

End condition: The IdP/SM-DP/MNO retrieved the private key corresponding to the request identity.

- **Identity/key provision for eUICC**

Start conditions:

a. The eUICC is registered at the SM-SR and ready for profile download.

b. An unpersonalised profile has been created by the SM-DP based on the profile description provided by the MNO.

c. The MNO has a request for a quantity of eUICC profiles.

d. The un-personalised profile has been validated on the target eUICC type using the un-personalised profile verification procedure.

Procedure:

1) the MNO provides the profile ordering to a selected SM-DP. The details of profile ordering process is shown in clause 3.5.3 of [b-GSMA SGP.02];

2) a personalized profile is created by the SM-DP using the data received from the MNO. In particular, SM-DP shall use the chosen international mobile subscription identity (IMSI) as the identity to complete the sign operation with the KMS as specified in the private key generation with KMIP to generate a private key for the chosen IMSI. The generated private key and ibPublicParameters in IBSysParams shall be included as keys in the profile;

3) the target profile is provisioned on the eUICC from the MNO. The details of profile the download and installation process can be seen in clause 3.5.4 of [b-GSMA SGP.02];

4) the target profile of the eUICC is enabled via SM-SR or via SM-DP and SM-SR. For the specific steps of profile enablement, see clause 3.5.6 or 3.5.7 of [b-GSMA SGP.02].

End condition: The target profile is enabled on the eUICC. The previously enabled profile is disabled. The EIS is up to date.

- **Identity and key provision for non-eUICC IoT devices**

Case 1: SecM has the capability to establish a TLS session with the IdP

Start conditions:

a. SecM has been registered at the AuC.

Procedure:

1) the SecM establishes a TLS session with the IdP and it shall successfully verify the validity of the IdP TLS certificate;

2) the SecM carries through a web authentication procedure with the IdP using PROV.ID and PROV.CRED;

3) the IdP chooses an identity assigned to the requesting device and completes the sign operation with the KMS as specified in the private key generation with KMIP to generate a private key for the chosen identity;

4) the IdP sends the assigned identity, the generated private key and the public parameters to the SecM through the TLS session;

5) the SecM stores the private key securely and the public parameters shall be protected from unauthorized change.

End condition: The target key is provisioned on the SecM.

The SecM and IdP shall follow the protocol defined in clause 5 and [IETF RFC 5408] to complete the identity and key provision procedure. In the response, the IBEPrivateKeyReply structure defined in [IETF RFC 5408] shall be replaced by IBPrivateKeyReply.

```
IBPrivateKeyReply ::= SEQUENCE SIZE (1..MAX) OF IBPrivateKey

IBPrivateKey ::= SEQUENCE {
        pkgIdentity         IBIdentityInfo OPTIONAL,
        pkgAlgorithm        OBJECT IDENTIFIER,
        pkgKeyData          IBPrivateKeyBlock,      --defined by pkgAlgorithm
        pkgOptions          SEQUENCE SIZE (1..MAX) OF PKGOption,
        ibSysParams         IBSysParams OPTIONAL
}

PKGOption ::=  SEQUENCE {
        optionID        OBJECT IDENTIFIER,
        optionValue     OCTET STRING
}
```

Case 2: SecM has no TLS implementation

Start conditions:

a. SecM has been registered at AuC.

Procedure:

1) the SecM generates the key encryption key (KEK) and encodes the key provisioning request (IBKeyProvRequest). The request includes the KEK, the provision ID (PROV.ID) and credential (PROV.CRED) which are encrypted by using the IdPs public key identified by IdP.ID. The encryption result is encoded as EncryptedMsg. It sends the encrypted request as the body of a HTTP POST request to the IdP;

2) the IdP decrypts the ciphertext by using the privacy key identified by IdP.ID in the request and checks the freshness of the timestamp or the correctness of the counter, or both. If the request does not pass these checks, the IdP shall return a response indicating a failure. The IdP further checks the correctness of PROV.ID and PROV.CRED with the AuC. If such a check fails, the IdP shall return a response indicating a failure. The IdP chooses an identity assigned to the requesting device and completes the sign operation with the KMS as specified in the private key generation with KMIP to generate a private key for the chosen identity;

3) the IdP encrypts the generated private key and, if necessary, the identity and the public parameters, which are encoded as IBKeyProvisionData, with the key encryption key (KEK) using the specified algorithm (keyProtAlg) conveyed in the request. The ciphertext is encoded as EncryptedMsg. The IdP sends the encrypted response as the body of the HTTP response to the SecM through;

4) the SecM decrypts the response and gets the assigned identity, the private key and the public parameters. It stores the private key securely and the public parameters shall be protected from unauthorized change.

End condition: The target key is provisioned on the SecM.

```
IBKeyProvisionRequest ::= SEQUENCE {
        version       INTEGER { v1(1) },
        timer         Time OPTIONAL,
        counter       INTEGER        OPTIONAL,
        identity      OCTET STRING,
        credential    OCTET STRING,
        keyProtAlg    OBJECT IDENTIFIER,
        kek           OCTET STRING
}
Time ::= CHOICE {
     utcTime          UTCTime,
     generalTime      GeneralizedTime
}
IBKeyProvisionResponse ::= SEQUENCE SIZE(1..MAX) OF IBKeyProvisionData
IBKeyProvisionData ::= SEQUENCE {
        identity          OCTET STRING OPTIONAL,
        ibSysParams       IBSysParams OPTIONAL,
        ibPrivateKey      IBPrivateKeyBlock
```

```
}
EncryptedMsg ::= SEQUENCE {
        encryptionAlgorithm EncryptionAlgorithmIdentifier,
        encryptedData       EncryptedData
}
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING
```

C.5 Identity and key revocation

If an identity must be disallowed in the IBC system for various reasons, e.g., because the owner of the identity has unsubscribed from the service or the corresponding private key has been compromised, the identity should be revoked and the corresponding private key may need to be destroyed for security reasons. If an identity is revoked, the identity shall be set in the revoked status. If an entry queries the status of a revoked identity, IdP/SM-DP/MNO shall return the correct value as defined in the OISP. To check the identity status more efficiently, an entity can retrieve the IRL from the IdP/SM-DP/MNO regularly and store it locally, and the entity can check with the fresh IRL to determine whether an identity is revoked without querying status online for each identity. For eUICC, the process to destroy the private key is possible by first disabling and then deleting the profile from the eUICC.

- Identity and key revocation for eUICC

Start conditions:

a. The target profile is enabled on the eUICC.

Procedure:

1) the MNO starts profile disablement via the SM-DP process. The details of the profile disabling process are shown in clause 3.5.8 of [b-GSMA SGP.02]. SM-DP shall set the identity in the revoked status;
2) the MNO starts the profile deletion process. For the specific steps in ISD-P deletion, see clause 3.5.10 of [b-GSMA SGP.02]. The SM-DP shall set the identity to revoked, and if the ISD-P deletion process is successful, the identity is also set to the deleted status. When an entity queries the status of an identity, the SM-DP shall respond appropriately according to the status record. Periodically, the SM-DP shall publish an identity status list of those with revoked identity during the period.

End condition: The target profile is disabled and deleted from the eUICC.

- **Identity and key revocation for non-eUICC IoT devices**

If an identity is revoked, the IdP shall set the identity to the revoked status. When an entity queries the status of an identity, IdP shall respond appropriately according to the status record. Periodically, the IdP shall publish an identity status list of those with revoked identity during the period.

The revocation trigger process and identity status maintenance lie outside the scope of this Recommendation.

- **Online identity status protocol**

With a large number of IoT devices connecting to a telecommunications operator, it may be necessary for an SM-DP, IdP or an IoT device to obtain timely information regarding the revocation status of an IoT device identity. In this Recommendation, an OISP is specified to enable the SM-DP, IdP or an IoT device to determine the current status of an identity with online queries. An OISP client issues a status request to an OISP responder and suspends acceptance of the identity in question until the

responder responds. The OISP shares similarity with the online certificate status protocol (OCSP) [IETF RFC 6960].

An OISP request contains the following data:

```
OISPRequest ::= SEQUENCE {
        version         INTEGER { v1(1) },
        identity        IBIdentityInfoSet
}
```

– version indicates the version of the protocol, which for this document is v1(1).

– identity is the OISP request.

```
IBIdentityInfoSet ::= SEQUENCE SIZE(1..MAX) OF IBIdentityInfo

IBIdentityInfo ::= SEQUENCE {
        domainName          IA5String OPTIONAL,
        domainSerial        INTEGER OPTIONAL,
        identityType        OBJECT IDENTIFIER OPTIONAL,
        identityData        OCTET STRING
}
```

– domainName is OPTIONAL and IA5String represents the URI [b-URI] or IRI [b-IRI].

– domainSerial is OPTIONAL and includes an INTEGER that defines a unique set of IBC public parameters in the event that more than one set of parameters is used by a single domain.

– identityType is OPTIONAL and contains an OBJECT IDENTIFIER that defines the format that the identityData field is encoded with. If this field is missing, default identity type is used.

– identityData is the data of the target identity.

Upon receipt of a request, an OISP responder checks whether the message is well formed and the request contains the information needed by the responder. If such a check fails, the OISP responder produces an error message; otherwise, it returns a definitive response according to the status of queried identities in the request.

```
OISPResponse ::= SEQUENCE {
        responseStatus      OISPResponseStatus,
        responseData        OISPResponseData OPTIONAL
}
```

– responseStatus indicates the processing status of the prior request.

– responseData is OPTIONAL and includes the response data of the request. If the value of responseStatus is one of the error conditions, the responseData field is not set.

```
OISPResponseStatus ::= ENUMERATED {
      successful(0),   -- Response has valid confirmations
      malformedRequest  (1),   -- Illegal confirmation request
      internalError     (2),   -- Internal error in issuer
      tryLater          (3),   -- Try again later
```

```
                        --(4) is not used
    unauthorized        (5)   -- Request unauthorized
}
OISPResponseData ::= SEQUENCE {
    version             INTEGER { v1(1) },
    producedAt          GeneralizedTime,
    hashAlgorithm       AlgorithmIdentifier OPTIONAL,
    tbsIdStatus         SEQUENCE OF SingleIdStatus,
    signatureAlgorithm  AlgorithmIdentifier OPTIONAL,
    signature           BIT STRING OPTIONAL,
    certs               [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}
```

– version MUST be v1(1) for this version of the basic response syntax.

– producedAt is the time at which the OISP responder signed this response.

– hashAlgorithm defines a hash algorithm to generate idHash in tbsIdStatus if such field exists. The field is optional and default value is OBJECT IDENTIFIER for SHA256 without parameters.

– tbsldStatus indicates the responses for each of the identity in a request.

– signatureAlgorithm is OPTIONAL and includes the algorithm that was used to sign the response.

– signature is computed upon the ASN.1 DER result from field producedAt to tbsIdStatus with the specified signature algorithm. This field is OPTIONAL and may not be set if the OISP client has other methods to guarantee the authenticity of the response. For example, the response is transmitted through a TLS secure channel between the client and responder.

– certs is OPTIONAL and indicates the certificate that helps the OISP client verify the responders signature. The structure of Certificate is defined in [IETF RFC 5280].

```
SingleIdStatus ::= SEQUENCE {
    idHash          OCTET STRING OPTIONAL,
    identityID      IBIdentityInfo OPTIONAL,
    identityStatus  IdentityStatus,
}
```

– idHash is OPTIONAL and includes the hash of the request identity. If identityID is too long, idHash can be used to represent the queried identity.identityID is OPTIONAL and contains the target identitys IBIdentityInfo field in the request.

– identityStatus indicates the status of the identity in the prior request.

```
IdentityStatus ::= CHOICE {
    good                [0]   IMPLICIT NULL,
    revoked             [1]   IMPLICIT RevokedInfo,
    unknown             [2]   IMPLICIT UnknownInfo,
    updated             [3]   IMPLICIT IBIdentityInfo,
    revokedAndDeleted   [4]   IMPLICIT RevokedInfo
```

}

UnknownInfo ::= NULL

– The "good" state indicates a positive response to the status inquiry.
– The "revoked" state indicates that the identity has been revoked, either temporarily or permanently and value is the revocation information.
– The "unknown" state indicates that the responder does not know about the certificate being requested.
– The "updated" state indicates that the identity has been updated and the value is a newly assigned identity for the queried identity.
– The "revokedAndDeleted" state indicates the identity has been revoked and the private key has been destroyed from the remote device.

```
RevokedInfo ::= SEQUENCE {
   revocationTime       GeneralizedTime,
   revocationReason    [0] EXPLICIT IRLReason OPTIONAL
}
IRLReason ::= ENUMERATED {
        unspecified             (0),
        keyCompromise           (1),
        pkgCompromise           (2),
        affiliationChanged      (3),
        superseded              (4),
        cessationOfOperation    (5),
        identityHold            (6),
                                -- value 7 is not used
        removeFromIRL           (8),
        privilegeWithdrawn      (9)
}
```

- **Identity revocation list**

Apart from using OSIP to respond the identity status queries, an entity such as IdP or SM-DP may publish a complete list of the revoked identities in a regular period, an IRL. To speed up the identity status checking process, a status checking entity with large storage may query the IRL and store it locally. The checking entity can determine whether an identity is acceptable for certain operations, such as network access authorization based on the IRL. If such identity is not in the IRL, then it is assumed that the identity is valid. To improve system efficiency, the IdP/SM-DP/MNO may just publish newly revoked identities since a specific time. It is called a delta IRL. A delta IRL contains the information of identities revoked since a complete IRL publication. The use of delta IRLs can significantly reduce the communication overhead and the processing time of IRLs. An IRL is similar to a certificate revocation list (CRL) [IETF RFC 5280].

The IRL is defined as

```
IdentityRevocationList ::= SEQUENCE {
     tbsIdentityList          TBSIdentityRevocationList,
```

```
    signatureAlgorithm      AlgorithmIdentifier OPTIONAL,
    signatureValue          BIT STRING OPTIONAL
}
```

– tbsIdentityList is the list of revoked identities with extra information, such as the revocation time.

– signatureAlgorithm defines the algorithm the issuer of IRL used to sign the list. This field is optional and is not present if signatureValue does not exist.

– signatureValue defines the value of signature generated by the issuer on the tbsIdentityList. This field is optional and not present if the request client has other means of guaranteeing that the retrieved list is authentic.

```
TBSIdentityRevocationList  ::=  SEQUENCE {
    version             INTEGER { v1(1) },
    issuer              Name,
    irlNumber           INTEGER OPTIONAL,
    deltaList           BOOLEAN OPTIONAL,
    thisUpdate          Time,
    nextUpdate          Time OPTIONAL,
    domainName          IA5String OPTIONAL,
    domainSerial        INTEGER OPTIONAL,
    revokedIdentities           SEQUENCE OF SEQUENCE  {
        identity        IBIdentityInfo,
        revocationDate          Time,
        irlEntryExtensions      Extensions OPTIONAL
    } OPTIONAL,
    irlExtensions               [0] EXPLICIT Extensions OPTIONAL
}
Name ::= CHOICE {--imported from [IETF RFC 5280]
    rdnSequence  RDNSequence
}
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET SIZE (1..MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
    type    AttributeType,
    value   AttributeValue
}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY -- DEFINED BY AttributeType
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
```

```
Extension ::=  SEQUENCE  {--imported from [IETF RFC 5280]
   extnID      OBJECT IDENTIFIER,
   critical    BOOLEAN DEFAULT FALSE,
   extnValue   OCTET STRING
               -- contains the DER encoding of an ASN.1 value
               -- corresponding to the extension type identified
               -- by extnID
}
```

– version indicates the version of IRL structure.

– issuer is the name of the entity that issues the IRL.

– irlNumber is the issuer number of current IRL. It starts from 0. For each complete IRL publication, the number increases by 1. It is optional.

– deltaList indicates if the current IRL is a delta IRL. The list contains only the information of identities revoked since a complete IRL publication indexed by irlNumber.

– thisUpdate specifies the time of generating this IRL.

– nextUpdate defines the time to generate next IRL. It is optional.

– domainName defines the IBC identity domain.

– domainSerial defines the IBC identity domain number.

– revokedIdentities is the revoked identity set.

  • identity is the data of revoked identity.
  • revocationDate is the time when the identity is revoked.
  • irlEntryExtensions defines possible extensions of revokedIdentity. Currently, no extension is defined.

– irlExtensions defines possible extensions for IRL. Currently, no extension is defined.

Annex D

Authentication

(This annex forms an integral part of this Recommendation.)

In this annex, four existing authentication protocols are extended to support IBC.

D.1 One-pass secret transport protocol

This protocol corresponds to secret key transport mechanism 2 in [ISO/IEC 11770-3]. It transfers a secret key, generated and encrypted and signed by entity A, from entity A to entity B with explicit key authentication from entity A to entity B and implicit key authentication from entity B to entity A. The explicit key authentication from entity A to entity B is achieved by entity A signing the encrypted secret and a time-variant parameter (TVP). The implicit key authentication from entity B to entity A is achieved by encrypting the secret with Bs ID, which implies only B can recover the secret. See Figure D.1.

To conduct the protocol, the following requirements shall be satisfied.

– Entity A has a signature private key *A.ib.prk* corresponding to its ID and related public parameters *A.ib.pubparam*;

– Entity B has a decryption private key *B.ib.prk* corresponding to its ID and related public parameters *B.ib.pubparam*;

– Entity A has access to an authenticated copy of entity Bs public parameter for encryption *B.ib.pubparam* and Bs ID;

– Entity B has access to an authenticated copy of entity As public parameter for signature *A.ib.pubparam* and As ID;

– The optional TVP shall be either a timestamp or sequence number. If timestamps are used, then entity A and B need to maintain synchronous clocks or use a trusted third party time stamp;

– A and B may share the same public parameters, i.e., *A.ib.pubparam = B.ib.pubparam*.

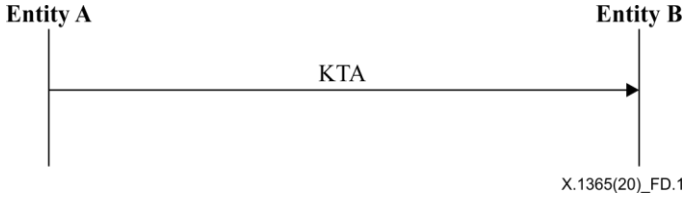

Figure D.1 – One-pass secret transport protocol

1) Entity A generates random secret *K* of the required length;

2) Entity A generates *BE*=**IBEnc**(*B.ib.pubparam, ID_B,* [*ID_A*]||*K*||*Text1*). Text1 can be empty, and *ID_A* is optional if entity B has other means to get entity As ID;

3) Entity A generates *S*=**IBSign**(*A.ib.pubparam, ID_A, A.ib.prk,* [*ID_B*]||TVP||BE||*Text2*). Text2 can be empty, and *ID_B* is optional if entity B knows the ID used *ID_B* for encryption;

4) Entity A generates token KTA=[*ID_B*]||TVP||BE|| *Text2*|| S||*Text3*;

5) When the TVP is a timestamp, entity B checks whether the TVP is within the allowed time difference. Otherwise, entity B rejects the token;

6) If entity B can get *ID_A* by other means and TVP is a sequence number, entity B first checks whether the sequence number is larger than the one maintained for entity B. Otherwise, entity B rejects the token;

7) If entity B can get *ID_A* by other means, entity B verifies the signature *S* in KTA by **IBVerify**(*A.ib.pubparam, ID_A,* [*ID_B*]||TVP||BE||*Text2,* S). If the signature is invalid, entity B rejects the token;

8) Entity B decrypts *BE* by *[ID_A]||K||Text1*=**IBDec**(*B.ib.pubparam*, *ID_B, B.ib.prk,* BE);

9) If entity B can only get *ID_A* after step 8, entity B checks the freshness of TVP, if TVP is a sequence number. If TVP is not fresh, entity B rejects the token. Entity B further verifies the signature *S*. If the signature is invalid, entity B rejects the token;

10) If all the checks and verifications are passed, entity A and entity B use *K* to protect the following messages. Both entities can use a key derivation function (KDF) [b-IEEE 1363] to generate keys for encryption and message authentication.

NOTE 1 – The protocol can be converted to a unilateral entity authentication protocol by removing BE from the message signed by entity A and KTA. This modification becomes the one-pass entity authentication scheme defined in [b-ISO/IEC 9798-3].

NOTE 2 – The protocol can be converted to a bilateral entity authentication protocol by requiring that entity B returns *K* to entity A. Entity B is authenticated by entity A by showing it can recover *K*, which requires it to own the private key *B.ib.prk*.

NOTE 3 – The identity-based signcryption algorithms such as the BLMQ signcryption algorithm [b-Barreto], Chen-Malone-Lee signcryption algorithm [b-Chen] may be used to improve efficiency.

D.2 TLS-IBS

In this clause, another authentication protocol named TLS-IBS is specified. It is assumed that both the server side and the IoT device side are provisioned with identity-based credentials, which include an identity, a private key for signature and KMS public parameters (e.g., KPAK as defined in [IETF RFC 6507] as a computing parameter). The KMS public parameter structure definitions for the supported algorithms can be found in Annex B.

The TLS-IBS is developed based on [IETF RFC 7250]. Traditionally, TLS client and server exchange public keys endorsed by public key infrastructure (PKI) certificates. It is considered to be complicated and may cause security weaknesses with the use of PKI certificates. To simplify certificate exchange, using a raw public key in TLS has been specified in [IETF RFC 7250]. That is, instead of transmitting a full certificate in the TLS messages, only public keys are exchanged between client and server. However, an out-of-band mechanism for public key and identity binding is assumed. For IoT networks, TLS with a raw public key is particularly attractive, but binding identities with public keys might be challenging. The cost of maintaining a large table for identity and public key mapping at the server side incurs additional maintenance cost, e.g. devices have to pre-register with the server. To simplify the binding between the public key and the entity presenting the public key, a better way could be using IBC, such as the ECCSI public key specified in [IETF RFC 6507], for authentication. Different from ITU-T X.509 certificates and raw public keys, a public key in IBC takes the form of the entity's identity. This helps eliminate the necessity of binding between a public key and the entity presenting the public key.

When IBS is used as a raw public key for TLS, signature and hash algorithms are negotiated during the handshake. The handshake between the TLS client and the server follows the procedures defined in [IETF RFC 7250] and TLS 1.3 [IETF RFC 8446], but with the support of the IBS algorithms as the signature schemes.

In the following, the TLS-IBS protocol developed based on [IETF RFC 7250] and TLS 1.3 with ECCSI [IETF RFC 6507], IBS1 (Hess-IBS), IBS1 (Cha-Cheon-IBS) and SM9-IBS [ISO/IEC 14888-3] as the signature algorithms is specified as follows.

1) the IoT device sends ClientHello including extension key_share, signature_algorithms, server_certificate_type, and client_certificate_type to the server, indicating that it supports raw public key and IBS algorithms;

2) the server sends ServerHello including extension key_share, server_certificate_type, client_certificate_type, Certificate, CertificateRequest, CertificateVerify and Finished to the IoT device, indicating raw public key is supported, and includes its identity (ServerID), KMS parameters (OID, KMS parameters), in the certificate part. Data structures for the KMS parameters are defined in clause D.2.3. A signature generated with the private key owned by the server is included in the CertificateVerify message;

3) after verifying the identity and signature of the server, the IoT device sends its raw public key as Certificate, CertificateVerify, and Finished to the server. The IoT device includes its identity (ClientID) and KMS parameters (OID, KMS parameters) in the certificate area, which is the raw public key of the client. Data structures for the KMS parameters are defined in clause D.2.3. A signature generated with the private key of client is included;

4) the remaining steps are the same as TLS 1.3 in [IETF RFC 8446].

See Figure D.2.

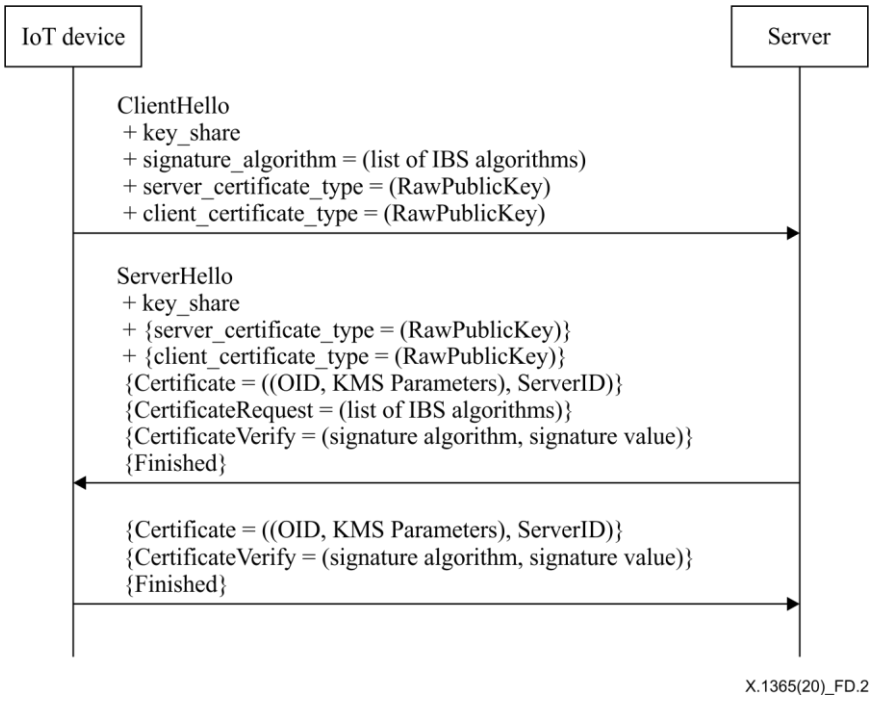

Figure D.2 – TLS-IBS

D.2.1 ClientHello

The ClientHello message format is the same as that specified in TLS 1.3 [IETF RFC 8446], but the values for signature algorithm need be extended for IBS.

The ClientHello message tells the server the types of certificate or raw public key supported by the client, and also the certificate types that the client expects to receive from the server. The ClientHello message includes desired IBS algorithms based on the order of client preference. In TLS 1.3, a data structure named SignatureScheme is defined for signature algorithms. To support the IBS algorithm, it has to be extended as follows:

```
enum {
    ...
  /* IBS signature algorithm */
    eccsi_sha256 (0x0704),
    ibs1_sha256(0x0705)
```

```
        ibs2_sha256(0x0706)
        sm9_ibs_sm3(0x0707)
      /* Reserved Code Points */
      private_use (0xFE00..0xFFFF),
      (0xFFFF)
  } SignatureScheme;
```

Details of the code points for the extended signature algorithms can be found in the TLS registry [b-IANA TLS REG].

D.2.2 ServerHello

The ServerHello message format is the same as that specified in TLS 1.3 [IETF RFC 8446]. The SignatureScheme is extended in the same way as in Client_Hello.

D.2.3 Server certificate

For the server certificate, a certificate structure is defined as RawPublicKey in [b-IEF RFC 7250]. As in [IETF RFC 7250], a data structure subjectPublicKeyInfo is used to specify the raw public key and its cryptographic algorithm. Within the subjectPublicKeyInfo structure, two fields, algorithm and parameters, are defined. The algorithm specifies the cryptographic algorithm used with a raw public key, which is represented by OIDs; and the parameters field provides necessary parameters associated with the algorithm. The identity of the server should be in the subjectPublicKey part.

NOTE – The identity should follow the format defined in Appendix I.

```
subjectPublicKeyInfo ::= SEQUENCE {
        algorithm                   AlgorithmIdentifier,
        subjectPublicKey                    BIT STRING
}
AlgorithmIdentifier ::= SEQUENCE {
        algorithm           OBJECT IDENTIFIER,
        parameters          ANY DEFINED BY algorithm OPTIONAL
}
```

When using an IBS algorithm, an identity is used as a raw public key, which can be converted to an OCTET string. Therefore, the certificate and subjectPublicKey structure can be reused without changes.

The algorithm field in AlgorithmIdentifier structure is the object ID of the IBS algorithm used. Besides that, it is necessary to tell the peer the set of public parameters used by the signer. The information can be carried in the payload of the parameters field in AlgorithmIdentifier. Corresponding to the algorithms above, the public parameter structures are ECCSIPublicParameters, BFPublicParameters, BFPublicParameters and SM9PublicParameters, respectively, as defined in Annex B.

To support IBS algorithms over TLS protocol to generate message CeritificateVerify, a data structure for signature value need to be defined.

- A data structure for ECCSI is defined as follows (based on [IETF RFC 6507]):

```
ECCSI-Sig-Value ::= SEQUENCE {
   r INTEGER,
   s INTEGER,
   pvt  OCTET STRING
```

}

where pvt (as PVT defined in [IETF RFC 6507]) is encoded as 0x04 || x-coordinate of [v]G || y-coordinate of [v]G.

- A data structure for IBS1 is defined as follows:

```
IBS1-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s ECPoint
  }
ECPoint ::= OCTET STRING as defined in [IETF RFC 5480]
```

- A data structure for IBS2 is defined as follows:

```
IBS2-Sig-Value ::= SEQUENCE {
    r ECPoint,
    s ECPoint
  }
```

- A data structure for SM9-IBS is defined as follows:

```
SM9-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s ECPoint
  }
```

To use a signature algorithm with TLS, an OID for the signature algorithm needs to be provided. Table D.1 shows the basic information needed for the IBS signature algorithms to be used for TLS.

Table D.1 – Identity-based signature algorithms

Key type	Document	OID
ISO/IEC 14888-3 ibs-1	ISO/IEC 14888-3: IBS-1 mechanism	1.0.14888.3.0.7
ISO/IEC 14888-3 ibs-2	ISO/IEC 14888-3: IBS-2 mechanism	1.0.14888.3.0.8
SM9-IBS	ISO/IEC 14888-3: Chinese IBS mechanism	1.2.156.10197.1.302.1
Elliptic curve-based certificateless signatures for identity-based encryption (ECCSI)	Section 5.2 of [IETF RFC 6507]	1.3.6.1.5.5.7.6.29

D.2.4 Client certificate

To support IBS, the client certificate is extended in the same way as the server certificate.

D.3 EAP-TLS-IBS

In this clause, EAP-TLS authentication protocol is extended to support IBS. Both the network side and the UE side are provisioned with identity-based credentials, which include an identity, a private key for signature and KMS public parameters (e.g., KPAK as defined in [IETF RFC 6507]). See Figure D.3.

The EAP-TLS is modified as follows.

1) the same as EAP-TLS;
2) after receiving the EAP-response with the UE identity, ID_UE;
3) AU sends ID_UE to RSF for validation;
4) RSF validates the ID_UE based on the revocation list stored;
5) RSF sends the validation result back to the AU;

6) if ID_UE is valid, then AU sends the EAP-TLS start message to UE;

7-9) the same as those described in the aforementioned TLS-IBS;

10) EAP-Success.

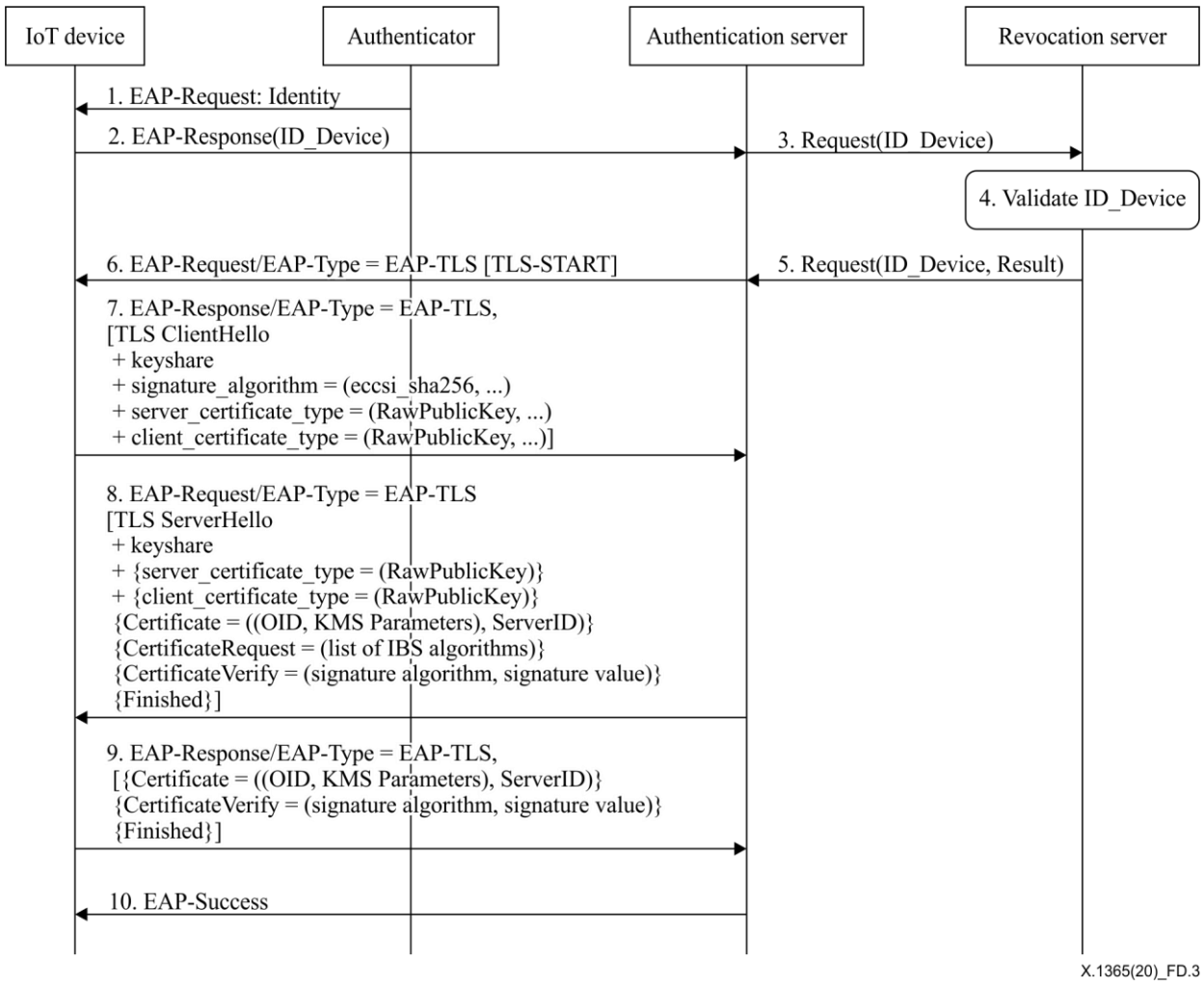

Figure D.3 – EAP-TLS-IBS

D.3.1 EAP-Request

The EAP-Request message format is the same as that specified in [IETF RFC 5216].

D.3.2 EAP-Response

The EAP-Response message format is the same as that specified in [IETF RFC 5216].

D.3.3 ClientHello

The ClientHello message format is the same as that in clause D.2.1.

D.3.4 ServerHello

The ServerHello message format is the same as that in clause D.2.2.

D.3.5 Server certificate

The server certificate format is the same as that in clause D.2.3.

D.3.7 Client certificate

The server certificate format is the same as that in clause D.2.4.

D.4 EAP-PSK-ECCSI

In this clause, EAP-PSK is extended to support one of the IBS algorithms, ECCSI, for authentication. Both UE and AU are provisioned with identity-based credentials, which include an identity, an SSK, a PVT, and a KPAK as defined in [IETF RFC 6507] as a computing parameter.

With the provisioned credentials, the UE and AU can derive symmetric keys based on static Diffie-Hellman by exchanging the identity information and the PVT, and then use the SSK owned by each entity. For example, a UE can derive a key after it receives the identity of the AU and its PVT, denoted as ID_AU and PVT_AU respectively, as follows:

K_UE = [SSK_UE](KPAK+[hash(G || KPAK || ID_AU || PVT_AU)]PVT_AU))

where G is a generation point on the elliptic curve used by the KMS to generate keys for UE and networks. It is provisioned to UE and AU by the KMS together with SSK, PVT and KPAK, etc. The use of the hash function can follow Annex A of [IETF RFC 6507].

Similarly, AU can also derive K_AU after it receives the identity and PVT from UE as follows:

K_AU = [SSK_AU](KPAK+[hash(G || KPAK || ID_UE || PVT_UE)]PVT_UE))

It can be proved that K_UE actually equals K_AU.

With the above properties, EAP-PSK can be used for mutual authentication as follows.

1) the UE sends an attach request to the AU and indicates that EAP-PSK shall be used for mutual authentication;
2) the AU verifies the authentication type and decides an authentication method;
3) the AU sends the first message about the EAP-PSK to the UE with an identity field that contains the ID_AU and PVT_AU, and also a random number RAND_S as required by EAP-PS;
4) the UE derives a symmetric key as K = [SSK_UE](KPAK+[hash(G || KPAK || ID_AU || PVT_AU)]PVT_AU)). The UE generates a random number RAND_P and further derives K = KDF(K, RAND_P, RAND_S). The UE derives an authentication key (AK) and key derivation key (KDK) based on [b-IETF RFC4764] for EAP-PSK;
5) the UE sends the second massage about EAP-PSK to the AU, which contains RAND_S, RAND_P, a MAC_P (MAC_P=CMAC-AES-128(AK, ID_P||ID_S||RAND_S||RAND_P)) for authentication, and an identity field that consists of ID_UE and PVT_UE;
6) the AU sends ID_UE to the RSF for validation;
7) the RSF validates the ID_UE according to its revocation list;
8) the RSF sends the validation results back to the AU;
9) if the ID is valid, then the AU derives a symmetric key as K = [SSK_AU](KPAK+[hash(G || KPAK || ID_UE || PVT_UE)]PVT_UE)). The AU further derives K = KDF(K, RAND_P, RAND_S). The AU derives an AK and KDK based on [IETF RFC 4764] for EAP-PSK. The AU authenticates the UE based on the MAC_P received from the message. The AU further derives a session key base on RAND_P and the KDK;
10) the AU sends the third message about the EAP-PSK to the UE with a MAC_S (MAC_S=CMAC-AES-128(AK, ID_S||RAND_P)) for authentication and other fields required by the EAP-PSK;
11) the UE authenticates the AU with MAC_S received and derives a session key with RAND_P and KDK derived previously;
12) the UE sends the last message about the EAP-PSK to the AU to finish the EAP-PSK authentication procedure.

See Figure D.4.

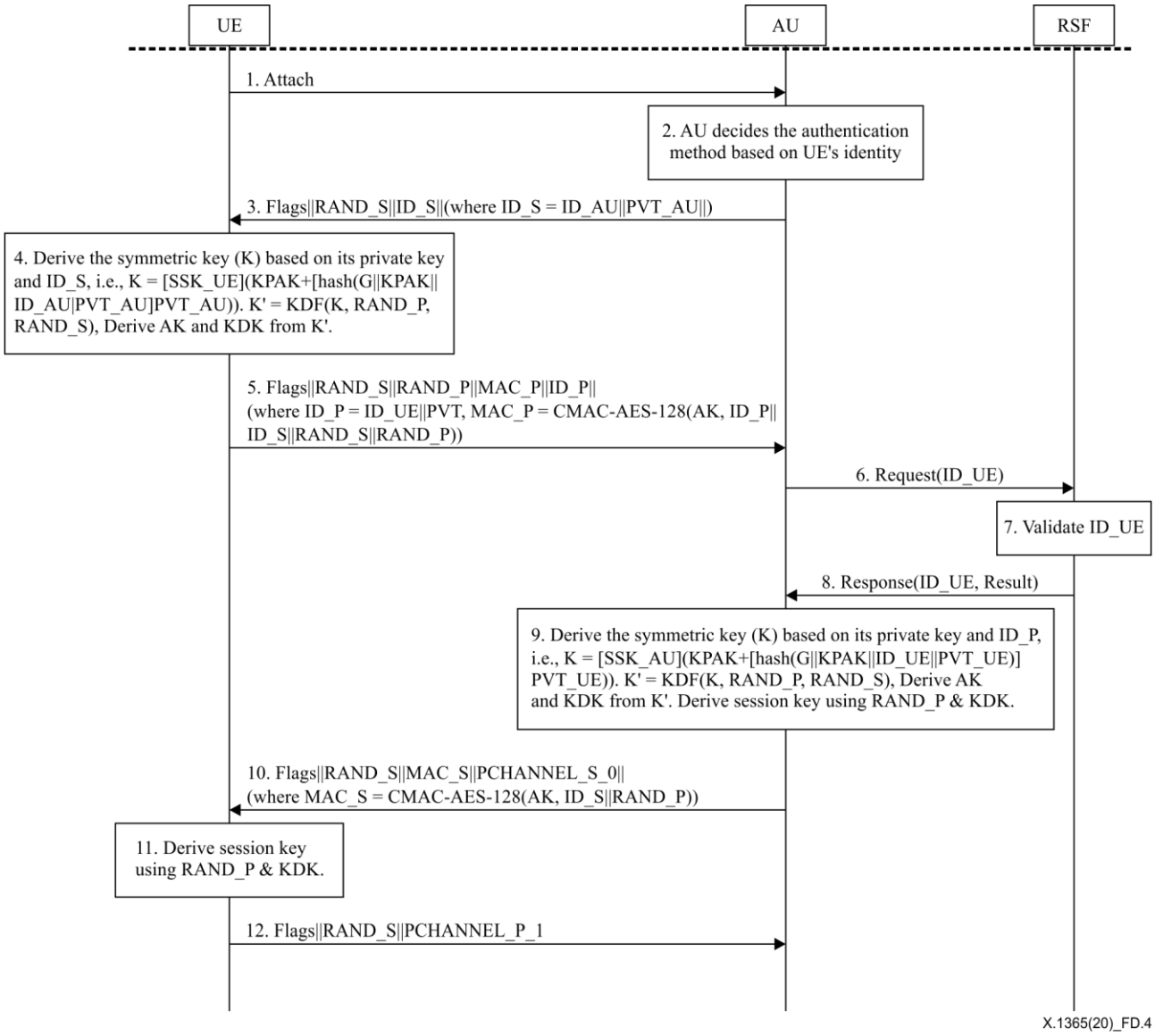

Figure D.4 – EAP-PSK--ECCSI

D.4.1 Attach

This message imitates the authentication procedure.

D.4.2 EAP-PSK--ECCSI first message (message 3 in Figure D.4)

The first EAP-PSK--ECCSI message is sent by the server to the peer. The format is as follows.

The first EAP-PSK--ECCSImessage consists of:

A 1-byte Flags field

A 16-byte random number: RAND_S

A variable length field that conveys the servers NAI: ID_S. The length of this field is deduced from the EAP length field. The length of this NAI must not exceed 966 bytes. This restriction aims to avoid fragmentation issues.

Figure D.5 shows an example format of the first message about the EAP-PSK.

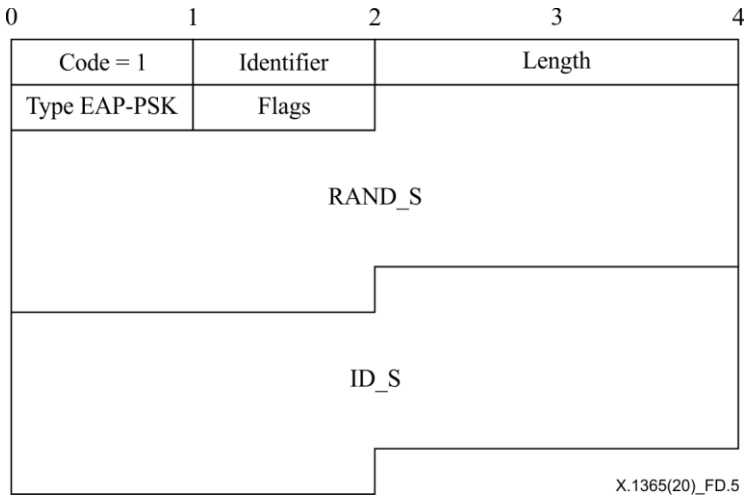

Figure D.5 – Format of EAP-PSK

To support IBC-based EAP-PSK authentication, the ID_S for the protocol of EAP-PSK is used to carry the ID_AU and PVT_AU. ID_S and PVT_AU are carried in the tag, length and vector (TLV) data structure, wherein the first octet carries a tag indicator, the second a length field, indicating the length of the field followed. The vector field carries the value.

Table D.2 defines the TLV for ID and PVT used with the EAP-PSK.

Table D.2 – Tag, length and vector definition for identity and public verification token

	Tag	Length	Value
Identity	1	Variable (≤255)	Defined by service provider
PVT	2	65	Number in hexadecimal

Figure D.6 shows the format of EAP-PSK--ECCSI message carrying the identity and PVT within the ID_S field.

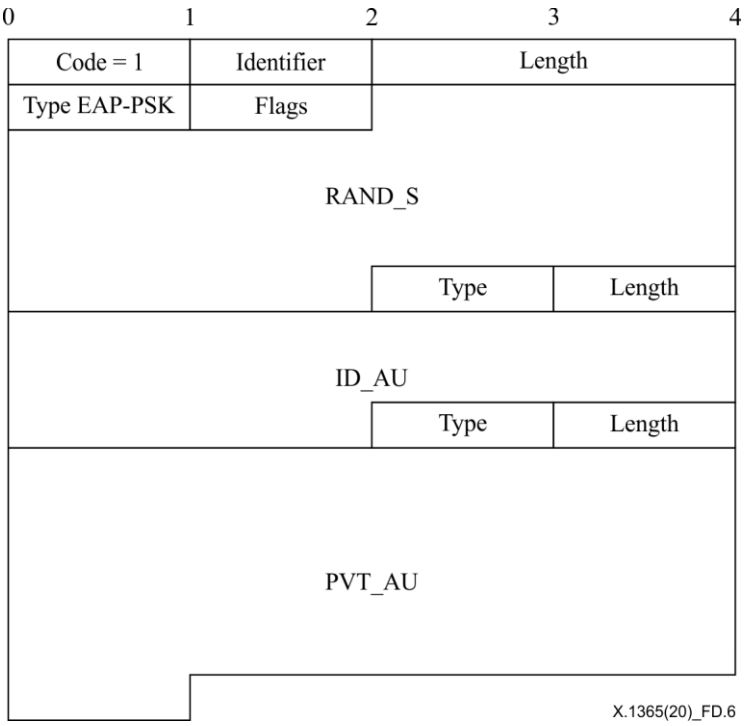

Figure D.6 – Message format for the EAP-PSK--ECCSI

D.4.3 EAP-PSK--ECCSI second message (message 5 in Figure D.4)

The second EAP-PSK-ECCSI message is sent by the peer to the server. The format consists of:

– a 1-byte flags field;

– the 16-byte random number sent by the server in the first EAP-PSK--ECCSI message (RAND_S) that serves as a session ID;

– a 16-byte random number: RAND_P;

– a 16-byte media access control (MAC): MAC_P;

– a variable length field that conveys the peers NAI: ID_P. The length of this field is deduced from the EAP length field. The length of this NAI must not exceed 966 bytes.

Similarly, the ID_S field of EAP-PSK is used to carry the ID_UE and PVT_UE field. Figure D.7 shows the format of the second EAP-PSK message.

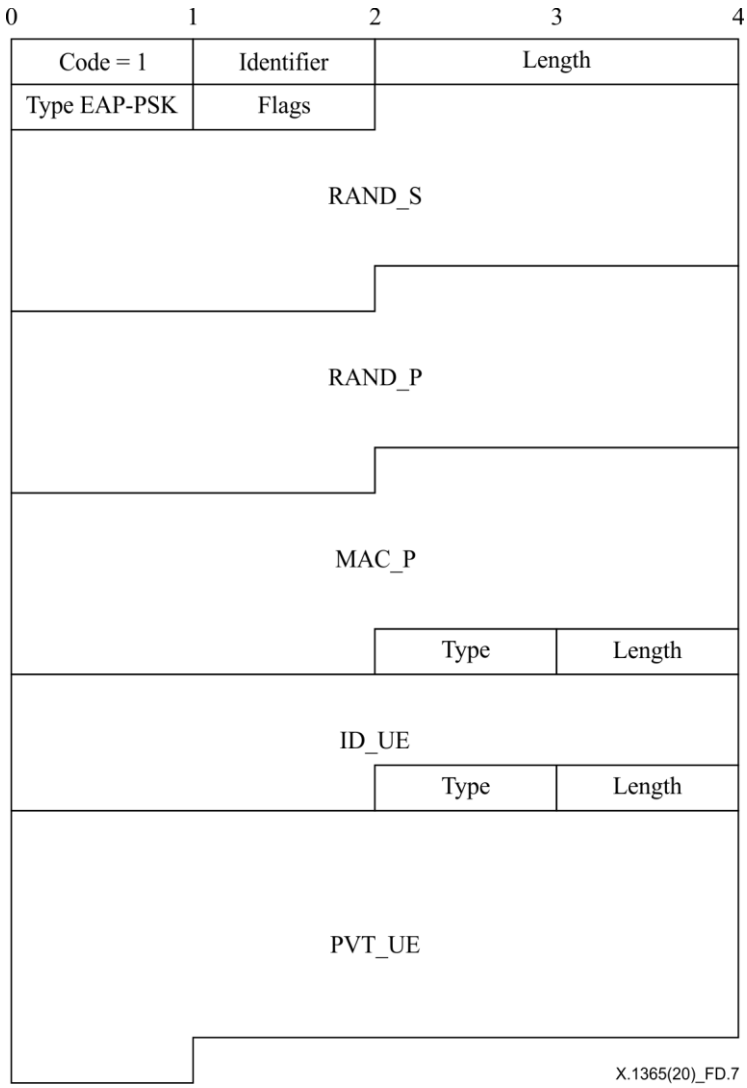

Figure D.7 – Message format for the second message about EAP-PSK--ECCSI

D.4.4 EAP-PSK--ECCSI third message (message 10 in Figure D.4)

The third EAP-PSK--ECCSI message is sent by the server to the peer. The format is the same as that presented in [IETF RFC 4764].

D.4.5 EAP-PSK--ECCSI fourth message (message 12 in Figure D.4-1)

The fourth EAP-PSK-ECCSI message is sent by the peer to the server. The format is the same as that presented in [IETF RFC 4764].

Appendix I

Identity naming

(This appendix does not form an integral part of this Recommendation.)

The ID in an IoT application can be the ID of a terminal or the ID of an IoT platform. The ID is a name that serves the purpose of identification. An ID is a handy representation of the object and allows the object, for example, in a database or in communication protocols to be referenced or addressed. In order to fulfil this purpose, IDs must be unique, or the ID is unique in an independent system. For example, the postal code is unique in a country, the uniqueness of the ID is given inside a certain scope. In addition, an ID is not only for a single object, but also for a group of objects, which enables uniform management and operation for this group.

OIDs [b-ITU-T X.660], [b-ITU-T X-Sup.31] are jointly developed by ISO/IEC and ITU-T, and have many characteristics. An OID has a hierarchical tree structure, which can flexibly extend its layers and the length of the IDs. An OID corresponds to a node in the OID tree, which is able to identify anything (physical or virtual, device or non-device), and is able to connect them with global information and communication infrastructures. The root of the tree contains the following three arcs: 0 (ITU-T), 1 (ISO) and 2 (joint-iso-itu-t). Each node in the tree is represented by a series of integers separated by periods, corresponding to the path from the root through the series of ancestor nodes, to the node. Each level of registration authority ID should be allocated by the upper level registration authority. For example, the OID denoting China National IC Card Registration Center, 1.2.156.20005 is allocated by 1.2.156 (ISO.member.china), the OID of China National OID Registration Center.

A complete OID would be a combination of registration authority ID and entity ID, and these two constituent parts are separated by a period, as shown in Figure I-1. If the company has registered an OID by the upper level registration authority, only the entity ID needs to be designed.

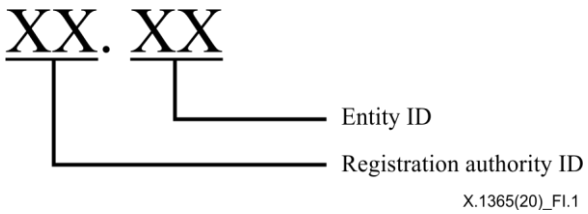

Figure I.1 – Structure of complete OID for objects

For example, the entity ID shall have the structure listed in Table I.1.

Table I.1 – Detailed information of the entity ID

Byte	Constituent part	Interpretation
1	version and reserved	4 bits for version of the entity ID, and 4 bits for reserved digits for the future
2	business	business type
3~11	expired time	invalid time of the identity, 5 bytes for issuing time in Unix time, and 4 bytes for validity period in seconds
12	type	value 0 for meaningless number, 1 for MAC, and 2 for IMSI
13	length (value *l*)	the size of the value part (in bytes), 6 for MAC and 8 for IMSI
14~13+*l*	value	individual identification number

The entity ID is 19 bytes long when using MAC as the individual identification number and 21 bytes long when using IMSI. An IMSI is usually presented as a 15-digit number or shorter, and the first digit is not zero except the test network [b-ITU-T E.212]. Padding zeroes before the IMSI to 16 digits and using 4 bits for one digit, 8 bytes is enough for an IMSI.

The IoT platform maintains a list for addressing. When a terminal device registers for the first time, the platform will add a row, which contains both the ID and the IP address of the device. By searching the ID of a device in the list, the IP address corresponding to the device can be obtained. See Table I.2.

Table I.2 – An example for the list for addressing

Identifier	IP address
1.2.9c.4e25.10.1.5b3e408003c26700.1.6.38B1DBC3156F	192.168.0.1

Appendix II

KMIP extensions to support IBC

(This appendix does not form an integral part of this Recommendation.)

The KMIP can be extended as follows to support required IBC operations with KMS, in particular the system initialization with KMIP and the private key generation with KMIP operations as defined in clauses C.1 and C.4 respectively.

The request payload for create key pair is composed as in Table II.1

Table II.1 – Request Payload

Object	Required	Description
Private key template-attribute	Yes	Specifies the attributes when the IBSetup function generates *ib.msk* and *ib.pubparam.*

Private key template-attribute shall include the attributes listed in Table II.2.

Table II.2 – Private key template-attribute

Object	Required	Encode	Description
Cryptographic algorithm	Yes	Enumeration, see Table II.3	Specifies the IBSetup function.
Cryptographic length	No	Integer	Specifies the bit length of the characteristic of the prime field on which the elliptic curve is based.
Cryptographic usage mask	Yes	Integer	Specifies the usage of the *ib.msk* which shall be Sign for key generation. IBExtract essentially is a signing process.
Cryptographic Domain Parameters	Yes	Object	Specifies more parameters for choosing system parameters, such as the elliptic curve used.
Cryptographic Parameters	Yes	Object	Specifies other functions, such as a hash function, which shall be used with the IBExtract functions.

Cryptographic algorithm shall be one of the values listed in Table II.3.

Table II.3 – Cryptographic algorithm (key generation)

Name	Value
IBC-KGA-BB1	00000030
IBC-KGA-BF	00000031
IBC-KGA-ECCSI	00000032
IBC-KGA-SK	00000033
IBC-KGA-SM9	00000034

Cryptographic length shall be a value equal or greater than 110.

Cryptographic usage shall be set as 00000001 (Sign).

Cryptographic domain parameters shall include the attributes listed in Table II.4.

Table II.4 – Cryptographic domain parameters

Object	Required	Encode	Description
QLength	No	Integer	Specifies the bit length of the order of the group from which *ib.msk* is chosen.
Recommended curve	Yes	Enumeration, see Table II.5	Specifies the curve used.
Pairing type	No	Enumeration, see Table II.6	If used, specifies the pairing in an identity-based algorithm.
Domain name	No	TEXT STRING	Specifies a unique name for the generated system parameters *ib.pubparam.*
Domain serial	No	INTEGER	Specifies a version number for the generated system parameters *ib.pubparam*.

Recommended curve shall be one of the values listed in Table II.5.

Table II.5 – Recommended curve

Name	Value
IBC-CURVE-SS1	00000070
IBC-CURVE-SS2	00000071
IBC-CURVE-BN-254-1	00000072
IBC-CURVE-BN-256-1	00000073
IBC-CURVE-BN-256-2	00000074
IBC-CURVE-BN-382-1	00000077
IBC-CURVE-BLS-12-381-1	0000007A
IBC-CURVE-BLS-12-442-1	0000007B
IBC-CURVE-BLS-12-455-1	0000007C
IBC-CURVE-BLS-12-461-1	0000007D
IBC-CURVE-KSS-16-340-1	0000007E
IBC-CURVE-KSS-18-348-1	0000007F

Pairing type shall be one of the values listed in Table II.6.

Table II.6 – Pairing type

Name	Value
Weil-Pairing	00000001
Tate-Pairing	00000002
Optimal-Ate-Pairing	00000003

Cryptographic parameters shall include the attributes listed in Table II.7.

Table II.7 – Cryptographic parameters

Object	Required	Encode	Description
Hashing algorithm	Yes	Enumeration, see Table II.8	Specifies the hash function which shall be used with the key generation function.
Private key group	No	Enumeration, see Table II.9	Specifies in which group the private key is generated if a pairing is used.

Hashing algorithm shall be one of the values listed in Table II.8.

Table II.8 – Cryptographic algorithm (hash)

Name	Value
SHA224	00000040
SHA256	00000041
SHA384	00000042
SHA512	00000043
SHA3-224	00000044
SHA3-256	00000045
SHA3-384	00000046
SHA3-512	00000047
SM3	00000048

Private key group shall be one of the values lited in Table II.9.

Table II.9 – Private key group

Name	Value
IBC-PRK-GROUP1	00000001
IBC-PRK-GROUP2	00000002
IBC-PRK-TWOGROUPS	00000003

The create key pair response payload is composed as in Table II.10.

Table II.10 – Response payload

Object	Required	Description
Private key unique identifier	Yes	The unique identifier of the newly created private key object which can be used to access *ib.msk*. The identifier is encoded as a text string.
Public key unique identifier	Yes	The unique identifier of the newly created public key object which can be used to access *ib.pubparam*. The identifier is encoded as a text string.

The request payload for get operation is composed as in Table II.11.

Table II.11 – Request payload

Object	Required	Description
Public key unique identifier	Yes	The unique identifier of the public key object which can be used to access *ib.pubparam*. The identifier is encoded as a text string.

The get response payload is composed as in Table II.12.

Table II.12 – Response payload

Object	Required	Description
Object type	Yes	Type of object
Unique identifier	Yes	The unique identifier of the object
Public key	Yes	A public key structure encapsulating the data of the IBC public parameters *ib.pubparam*

The unique ID shall be same as the public key unique ID sent in the get request payload.

Object type shall be 00000003 (public key).

The key block in the public key field is composed as in Table II.13.

Table II.13 – Key block in the public key field

Object	Required	Encode	Description
Key format type	Yes	Enumerate, see Table II.14.	Specifies the format of the key value.
Key compression	No	Enumerate.	Specifies if the key value should be compressed.
Key value	Yes	Transparent key structure for IBC public parameters.	A newly defined transparent key structure for IBC public key.
Cryptographic algorithm	Yes	Enumerate, see Table II.15.	Same as create key pair request payload

Key format type shall be the value in Table II.14.

Table II.14 – Key format type

Name	Value
Transparent IBC public parameters	00000016

Key compression shall be either 00000001 (uncompressed) or 00000002 (compressed prime).

Key value shall have attributes listed in Table II.15:

Table II.15 – Key value

Object	Required	Encode	Description
P	No	Big Integer	For curves based on a prime field, P is the characteristic (p) of the prime field.
Q	No	Big Integer	Q is the order of the point subgroup (*G*1) in which the cryptographic operations are computed.
J	No	Big Integer	J is the cofactor such that J*Q = X-1, where X is the order of point group of the specified curve.
P1 STRING	Yes	BYTE STRING	For pairing-based algorithms, P1 is the generator of group *G*1 of pairing. For the non-pairing-based algorithm, P1 is a generator of the working point subgroup.
P2 STRING	No	BYTE STRING	For pairing-based algorithms, P2 is the generator of group *G*2 of pairing.

Table II.15 – Key value

Object	Required	Encode	Description
sP1 STRING	No	BYTE STRING	sP1 is the scalar result of [*ib.msk*]P1 or the scalar result of an integer component of *ib.msk* with P1
sP2 STRING	No	BYTE STRING	For pairing-based algorithms, sP2 is the scalar result of [*ib.msk*]P2 or the scalar result of an integer component of *ib.msk* with P2
sP3 STRING	No	BYTE STRING	From some pairing-based algorithm (particularly algorithms using the BB1 key generation function), sP3 is the scalar result of another integer component of ib.msk with P1.
Public Pairing STRING	No	BYTE STRING	For some pairing-based algorithms, public pairing is the result of pairing(P1, [s]P2) or pairing([s]P1, P2) or pairing(P1, P2), where s is *ib.msk*, for algorithms like SM9, SK-KEM or SK-KEM or ([s1]P1, [s2]P2) for BB1-KEM, where s1, s2 are integer components of *ib.msk*.

The new tag definitions are listed in Table II.16.

Table II.16 – Tag definitions

Object	Tag Value
Pairing type	420100
Private key group	420101
Domain name	420102
Domain serial	420103
P1 STRING	420104
sP1 STRING	420105
P2 STRING	420106
sP2 STRING	420107
sP3 STRING	420108
Public pairing STRING	420109

The request payload for sign is composed as in Table II.17.

Table II.17 – Request payload for sign

Object	Required	Description
Unique identifier	No	The unique identifier of the managed cryptographic object that is the *ib.msk* key to use for the IBExtract operation. If omitted, then the ID placeholder value shall be used by the server as the unique identifier.
Cryptographic parameters	No	Cryptographic parameters may specify the group of which the private key shall be generated.
Data	Yes	Data specifies the identity value from which the private key shall be extracted.

Cryptographic parameters shall include the attributes listed in Table II.18.

Table II.18 – Cryptographic parameters

Object	Required	Encode	Description
Private key group	No	Enumeration, see Table II.9.	Specify the group (*G*1 or *G*2) of which the private key shall be generated.

Bibliography

[b-ITU-T E.101] Recommendation ITU-T E.101 (2009), *Definitions of terms used for identifiers (names, numbers, addresses and other identifiers) for public telecommunication services and networks in the E-series Recommendations*.

[b-ITU-T E.212] Recommendation ITU-T E.212 (2016), *The international identification plan for public networks and subscriptions*.

[b-ITU-T X.509] Recommendation ITU-T X.509 (2019), *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.

[b-ITU-T X.660] Recommendation 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*.

[b-ITU-T X.1361] Recommendation ITU-T X.1361 (2018), *Security framework for the Internet of things based on the gateway model*.

[b-ITU-T X-Sup.31] ITU-T X-series Recommendations – Supplement 31 (2017), *ITU-T X.660 – Supplement on guidelines for using object identifiers for the Internet of things*.

[b-ITU-T Y.2720] Recommendation ITU-T Y.2720 (2009), *NGN identity management framework*.

[b-ITU-T Y.4000] Recommendation ITU-T Y.4000/Y.2060 (2012), *Overview of the Internet of things*.

[b-ITU-T Y.4100] Recommendation ITU-T Y.4100/Y.2066 (2014), *Common requirements of the Internet of things*.

[b-ISO/IEC 9798-3] ISO/IEC 9798-3:2019. *IT Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques*.

[b-ETSI TR 118 508] ETSI TR 118 508 V1.0.0 (2014), *Analysis of Security Solutions for the oneM2M System*.
<https://www.etsi.org/deliver/etsi_tr/118500_118599/118508/01.00.00_60/tr_118508v010000p.pdf>

[b-ETSI TS 133.501] ETSI TS 133 501 V15.2.0 (2018), *5G; Security architecture and procedures for 5G system (3GPP TS 33.501 version 15.1.0 Release 15)*.
<https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.01.00_60/ts_133501v150100p.pdf>

[b-GM/T 0044.2] GM/T 0044.2-2016, *Identity-based cryptographic algorithms SM9 – Part 2: Digital signature algorithm*.

[b-GSMA SGP.02] GSMA Official Document SGP.02 Version 3.1 (2016), *Remote Provisioning Architecture for Embedded UICC – Technical Specification*.

[b-IANA TLS REG] Internet Assigned Numbers Authority (IANA), *Transport Layer Security (TLS) Parameters*. Website available, last viewed 2019-07-12.
<https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml>

[b-IEEE 1363] IEEE 1363-2000, *IEEE Standard Specifications for Public-Key Cryptography*.

[b-IEEE P1363.3] IEEE P1363.3/D9 (May 2013), *IEEE Standard for Identity-Based Cryptographic Techniques using Pairings*.

[b-IETF RFC 3748] IETF RFC 3748 (2004). *Extensible Authentication Protocol (EAP)*.

[b-OASIS KMIP] OASIS (2016), *Key Management Interoperability Protocol Specification Version 1.3*. <http://docs.oasis-open.org/kmip/spec/v1.3/os/kmip-spec-v1.3-os.pdf>

[b-Barreto] Barreto, P. S. L. M., Libert, B., McCullagh, N., Quisquater, J.-J. (2005). Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Roy B. (ed.). *Advances in Cryptology – ASIACRYPT 2005, pp. 515-532. Lecture Notes in Computer Science*, vol. 3788. Berlin: Springer

[b-Chen] Chen, L., Malone-Lee, J. (2005). Improved identity-based signcryption. In: Vaudenay S. (ed). *Public Key Cryptography – PKC 2005, pp. 362-379. Lecture Notes in Computer Science*, vol. 3386. Berlin: Springer.

[b-Ducas] Ducas, L., Lyubashevsky, V., Prest, T. (2014). Efficient identity-based encryption over NTRU lattices. In: Sarkar P., Iwata T. (eds). *Advances in Cryptology – ASIACRYPT 2014, pp. 22-41. Lecture Notes in Computer Science*, vol. 8874. Berlin: Springer.

[b-Freeman] Freeman, D., Scott, M., Teske, E. (2010). A taxonomy of pairing-friendly elliptic curves. *J. Cryptol*. **23**, pp. 224–280.

[b-Galbraith] Galbraith, S.D., Paterson, K.G., Smart, N.P. (2008). Pairings for cryptographers. *Discrete Appl. Math*., **156**, pp. 3113-3121.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks, open system communications and security**
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2020