3GPP2 S.S0132-0

Version 1.0

Version Date: January 28, 2010

Femtocell Security Framework

EDITOR

Anand Palanigounder
QUALCOMM Incorporated
Tel: (+1) 858 845 0193
Email: apg@qualcomm.com

REVISION HISTORY

REVISION HISTORY		
1.0	*Initial Publication*	*January 28, 2010*

Table of Contents

1 Introduction

The network architecture for cdma2000®[1] Femtocell systems is defined in [1]. This document defines the Security Framework for Femtocell systems in cdma2000 networks.

In this document, several key words are used to signify the requirements. The key words “shall”, “shall not”, “should”, “should not” and “may” are to be interpreted as described in the TIA Engineering Style Manual.

2 Scope

This document defines the security requirements, security architecture and mechanisms for securely connecting Femtocell Access Points (or Femtocells) to the cdma2000 networks.

3 References

3.1 Normative References

[1] 3GPP2 X.S0059-0 000: “cdma2000 Femtocell Network: Overview”.

[2] 3GPP2 X.S0059-0 100: “cdma2000 Femtocell Network: Packet Data Network Aspects”.

[3] 3GPP2 X.S0059-0 200: “cdma2000 1x and IMS Network Aspects”.

[4] 3GPP2 S.R0126-0: “System Requirements for Femto Cell Systems”.

[5] 3GPP2 A.S0024-0: “Interoperability Specification (IOS) for Femtocell Access Points”.

[6] IETF RFC 4306: “Internet Key Exchange (IKEv2) Protocol”.

[7] IETF RFC 4303: “IP Encapsulating Security Payload (ESP)”.

[8] IETF RFC 3948: “UDP Encapsulation of IPsec ESP Packets”.

[9] 3GPP2 C.S0005-D: “Upper Layer (Layer 3) Signaling Standard for cdma2000 Spread Spectrum Systems”.

[10] 3GPP TS 33.203: “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Access security for IP-based services”.

[11] IETF RFC 5280: “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile” (obsoletes IETF RFC 3280).

[1] cdma2000® is the trademark for the technical nomenclature for certain specifications and standards of the Organizational Partners (OPs) of 3GPP2. Geographically (and as of the date of publication), cdma2000® is a registered trademark of the Telecommunications Industry Association (TIA-USA) in the United States

[12] IETF RFC 4945: "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX".

[13] IETF RFC 3447: "PKCS #1: RSA Cryptography Specifications Version 2.1".

[14] IETF RFC 4055: "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[15] IETF RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[16] The Broadband Forum TR-069: "CPE WAN Management Protocol v1.1", Issue 1 Amendment 2, December 2007

[17] IETF RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1".

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

[19] IETF RFC 3268: "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)".

[20] National Institute of Standards and Technology: "Secure Hash Standard", FIPS 180-2, With Change Notice 1 dated February 2004.

3.2 Informative References

<1> IETF RFC 4187: "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)".

4 Definitions and Abbreviations

4.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Femtocell Access Point: A radio access network element that supports one or more of the cdma2000 family of radio interfaces, operates in a limited geographic area in licensed spectrum, may operate over the public internet, and supports a limited number of simultaneous users in generally small environments such as a home.

Femtocell systems: A set of one or more femtocells and a set of core network elements to manage and support the use of those femtocells in accessing network services.

4.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AAA	Authentication, Authorization and Accounting Server
ACS	Auto-Configuration Server
AT	Access Terminal
CA	Certificate Authority
CAVE	Cellular Authentication and Voice Encryption
CPE	Customer Premises Equipment
ESP	Encapsulating Security Payload
EUI-64	Extended Unique Identifier – 64-bit
FAP	Femtocell Access Point
FCS	Femtocell Convergence Server
FMS	Femtocell Management System
FEID	FAP Equipment Identifier
FQDN	Fully Qualified Domain Name
HRPD	High Rate Packet Data
IKEv2	Internet Key Exchange version 2
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPsec	IP Security
MNO	Mobile Network Operator
MS	Mobile Station
MSC	Mobile Switching Center
OUI	Organizationally Unique Identifier

PKCS	Public Key Cryptography Standards
SA	Security Association
SeGW	Security GateWay
SIP	Session Initiation Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UMB	Ultra Mobile Broadband
VLR	Visited Location Register

5 Overview of the Security Architecture

5.1 Reference Model

Figure 1 is the femtocell security architecture reference model that identifies the common functional elements and the interfaces such that it is applicable to any cdma2000 femtocell systems (e.g., 1x, HRPD, UMB).

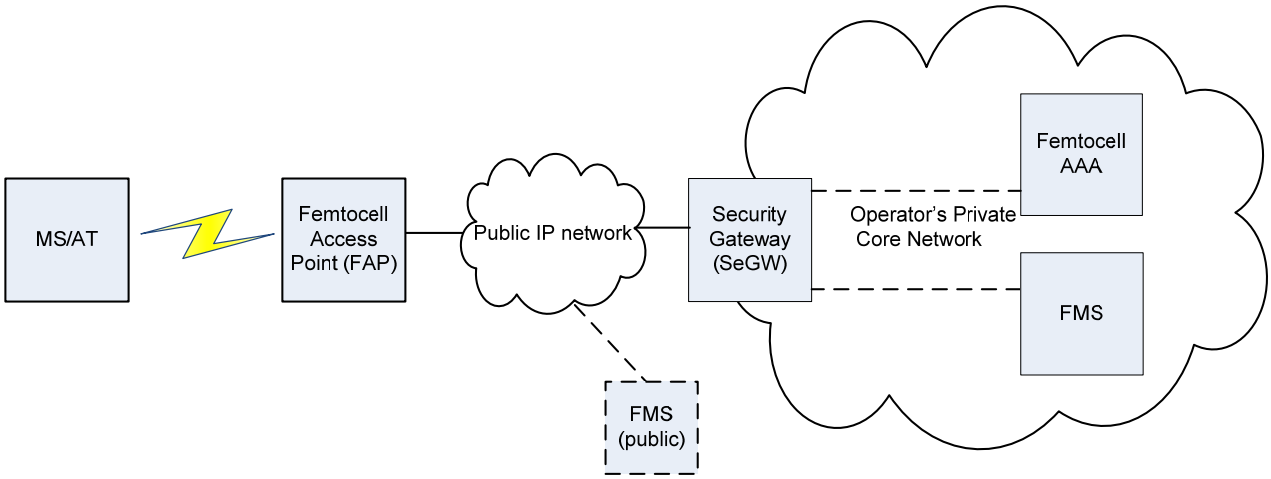

Figure 1. Femtocell Security Architecture Reference Model

The MS/AT uses the cdma2000 air interface to access services through a Femtocell Access Point (FAP). The FAP uses a Security Gateway (SeGW) to securely connect using an IP network to a cdma2000 operator's core network. Since the IP network (e.g., broadband connection) between the FAP and the SeGW is assumed to be un-trusted, the FAP shall be authenticated and authorized by the cdma2000 network before a FAP is allowed to provide service to the ATs. The Femtocell AAA is the entity in the cdma2000 network that has access to the authentication and authorization-related credential information that is required for securely operating the FAPs.

NOTE: In addition to authentication and authorization information, the Femtocell AAA may have additional profile information, but this is outside the scope of this document.

The functionalities of the FAP and how it interfaces to the cdma2000 network are defined in [1] . The overall Femtocell system requirements are defined in [4].

The Femtocell Management System (FMS) is a management server that is used to configure and monitor the operation of the FAPs using TR-069 protocol as defined in [16]. A FAP is considered a Customer Premises Equipment (CPE) and the FMS is the Auto-Configuration Server in the TR-069 management architecture. The FMS may also be capable of other management operations, e.g., installing software updates on the FAPs. The FMS is typically assumed to be located inside the operator's core network (i.e., reachable by the FAP only through the SeGW). However, in certain scenarios (e.g., the FAP is unable to connect to SeGW), the operator may have an FMS available on the public IP network (e.g., the internet) so that the FMS can connect to the FAP to re-initialize or diagnose the problem. In such cases, the FAP and the FMS in the public domain (public FMS) need to employ additional security measures to protect against the increased risk.

6 Security Features

6.1 FAP Device Identity

A FAP needs to be uniquely identified for authentication, authorization, device validation and verification purposes. This identity is referred to as FAP Equipment Identifier (FEID).

The FEID shall be globally unique. It shall be provisioned by the FAP manufacturer and be composed of a globally unique manufacturer identifier and an identity that is local to the manufacturer (e.g., serial number). The value of FEID shall remain fixed over the lifetime of the device, including across firmware updates.

The format defined by the Institute of Electrical and Electronics Engineers (IEEE) to identity devices using their hardware address meets the requirements for FAP device identity. Therefore, the FEID shall be compliant with Extended Unique Identifier – 64 or EUI-64 format (EUI-64)[1] defined by IEEE. The EUI-64 supports both 48-bit hardware address space (e.g., MAC address) and 64-bit hardware address space. The EUI-64 format of the FEID shall be formed from the device hardware address following the rules defined by IEEE.

[1] The "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY" are available at: <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

The OUI and the manufacturer assigned number of the FEID shall be encoded as hexadecimal digits value, including any leading zeros.

6.2 FAP Secure Environment

The security sensitive FAP credentials (e.g. private key of the FAP certificate, trusted CA certificate store) and cryptographic operations that make use of these credentials shall be isolated from unauthorized access and/or modification from other software modules inside the FAP and shall be protected against any outside probing. To comply with these security requirements, the FAP shall contain a Secure Environment, a logically separate entity within the FAP that provides secure storage and secure execution functionalities.

A Secure Environment shall be the component inside the FAP that stores sensitive information, such as the private key of the FAP certificate and executes security-critical cryptographic functions (such as the FAP authentication) that make use of this stored sensitive information. The private key of the FAP shall be stored inside the Secure Environment and shall not be exposed, be accessible or be used outside of the Secure Environment. The Secure Environment shall also perform cryptographic operations on behalf of the FAP such as during the secure start-up (boot-up) process, or device authentication to the operator's network.

The Secure Environment shall support a secure start-up process for the FAP (i.e., secure boot). The secure start-up process shall occur whenever the FAP is turned on or goes through a hard reset. The secure start-up process shall include the integrity validation of every component of the Secure Environment. Only successfully validated components shall be loaded or started. The Secure Environment may also perform integrity validation of FAP components outside of the Secure Environment (e.g., operating system and other software modules). This process is referred to as the Device Integrity Validation in this document.

6.3 Authentication

6.3.1 Authentication of FAP and network

In order to ensure that the FAP is connecting to the desired cdma2000 network, the FAP shall be able to authenticate the SeGW it is connecting to, based on a certificate chain that ends in a trusted root certificate stored in the FAP.

Before a FAP is allowed to connect to a cdma2000 network, the SeGW shall authenticate the FAP, based on a certificate chain that ends in a trusted root certificate stored in the SeGW. The mutual authentication between the FAP and the network is referred to as FAP device authentication.

6.4 Authorization of FAP

It is assumed that the FAP is authorized by the home system (e.g., Femtocell AAA or the SeGW operated by the home system). Before a FAP is allowed to provide service, it shall be possible for the

home network to authorize FAP service based on the FEID that is used for the device authentication. This is referred to as FAP subscription authorization.

6.5 Integrity Protection

After the successful FAP device authentication, integrity protection shall be applied to any subsequent traffic between the FAP and the SeGW.

6.6 Confidentiality Protection

After the successful FAP device authentication, it shall be possible to apply confidentiality protection to any subsequent traffic between the FAP and the SeGW.

6.7 FMS Security

When the Femtocell Management System (FMS) is inside the operator's core network, the FAP shall communicate to the FMS via the SeGW. In this case, the traffic between the FAP and the SeGW shall be protected using the IPsec tunnel. The traffic between the SeGW and the FMS in the operator's core network may be protected using network domain security mechanisms if the link between the SeGW and FMS is considered insecure by the operator. Alternatively, when the FMS is inside the operator's core network, a TLS tunnel (as specified in [17] or [18]) may be used to protect the traffic between the FAP and the FMS.

In certain scenarios (e.g., the FAP is unable to connect to SeGW), the operator may have an FMS available on the public IP network (e.g., the internet) for diagnosis and initial configuration of FAP. When the FMS is in the public network domain (as opposed to the FMS in the protected network domain and only reachable through the SeGW), the following requirements shall be met:

- FAP and the FMS shall be mutually authenticated.
- All traffic between the FAP and the FMS shall be integrity and confidentiality protected.

6.7.1 Security for File Transfer

Managing a FAP using a FMS may require file transfer between the FAP and a management server, such as for initial configuration, updating the FAP software, or monitoring FAP status. When the FAP needs to download a file, the FMS may provide the file directly or provide a link to the actual file for download.

The file transfer between the FAP and the FMS (or another server) shall be integrity protected. The FAP shall always verify the integrity of the file before accepting it for further operation.

7 Security Mechanisms

7.1 Device Integrity Validation

Upon power-up or hard reset, the FAP shall perform Device Integrity Validation before attempting connection to the SeGW and/or to the FMS. The Device Integrity Validation shall be based on one or more trusted reference value(s) and the Secure Environment of the FAP. The trusted reference values are stored securely inside the FAP (e.g., secure storage) and shall be protected against any unauthorized modification. The FAP is considered passing the Device Integrity Validation, if all components required for the trusted operation of the FAP are validated. Each necessary FAP component is validated by comparing the result of a measurement (e.g., cryptographic hash) of the component to the trusted reference value. If these values match, the component is successfully validated and can be loaded and/or started.

7.2 FAP Device Authentication

The FAP device authentication shall be between the FAP and the SeGW. The authentication shall be performed using IKEv2 [6] with certificates. A profile for IKEv2 is defined in section 7.7.

The FAP shall authenticate itself to the SeGW using the FAP certificate. The FAP certificate shall be identified by the FEID. The FAP certificate profile is defined in section 7.5.

The SeGW shall authenticate itself to the FAP using SeGW's certificate. The root certificate used to verify the SeGW certificate shall be stored in the Secure Environment. The root certificate(s) store shall be protected against any unauthorized modification. The SeGW certificate shall be identifiable using either a fully qualified domain name (FQDN) or the IP address (e.g., Domain Name Server is not available) of the SeGW. Optionally, the FAP may check the revocation status of the SeGW certificate using OCSP (Online Certificate Status Protocol) as specified in [15]. The SeGW certificate profile is defined in section 7.6.

> NOTE: It is acknowledged that for the authentication of legacy (i.e., pre-standards) FAP, IKEv2 procedures can be used to negotiate another method supported by IKEv2 such as EAP-AKA <1>. An informative call flow using EAP-AKA is shown in Annex B.

After the successful mutual authentication between the FAP and the SeGW using IKEv2, IPsec SA(s) shall be established to protect all the subsequent traffic between the FAP and the SeGW. A profile for IPsec ESP is defined in section 7.8.

7.2.1 Certificate based Device Authentication

The FAP and the SeGW use the IKEv2 certificate based mutual authentication as specified in [6]. The IKEv2 certificate based device authentication is described in this section.

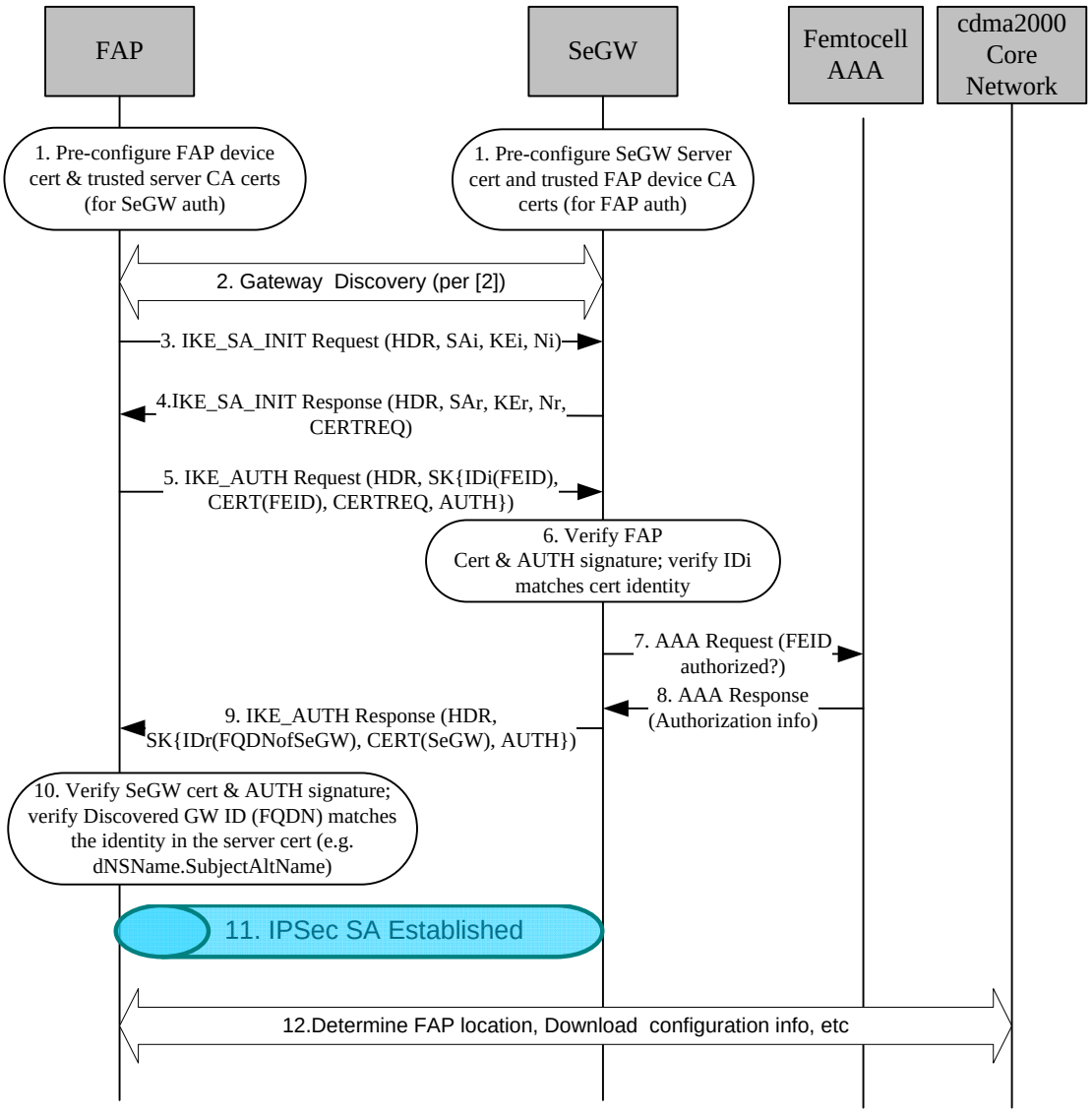

Figure 2. Certificate based FAP Device Authentication using IKEv2

1. FAP is configured with a device certificate during its manufacturing. The FAP device certificate is signed by a Certificate Authority (device certificate CA) trusted by the cdma2000 operator. The private key for the certificate is stored securely in the Secure Environment in the FAP. Similarly, the SeGW is configured with a server certificate. The private key of the SeGW server certificate is stored securely at the SeGW. The SeGW is also securely configured with a list of trusted CA certificates corresponding to the FAP device certificate CAs. The FAP is also securely configured with a list of trusted CA certificates corresponding to the server certificates that the FAP will accept for the SeGW.

2. Upon FAP power-up, using the SeGW discovery procedures specified in [2], the FAP determines the FQDN/IP address of the SeGW specific to the desired cdma2000 network.

3. The FAP initiates the IKEv2 exchange with the discovered SeGW, known as IKE_SA_INIT exchange, by issuing an IKE_SA_INIT Request to negotiate cryptographic algorithms, exchange nonces and perform a Diffie-Hellman exchange with the SeGW. In addition, using the NAT Traversal procedures outlined in section 2.23 of [6], the initiator includes NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads to negotiate support for UDP encapsulation.

4. The SeGW responds with IKE_SA_INIT Response by choosing a cryptographic suite from the initiator's offered choices, completing the Diffie-Hellman and the nonce exchange with the FAP. In addition, the SeGW includes the list of FAP CA certificates that it will accept in the CERTREQ payload. For successful FAP authentication, the CERTREQ payload has to contain at least one CA certificate that is in the trust chain of the FAP certificate. At this point in the negotiation, IKE_SA_INIT exchange is complete and all but the headers of all the messages that follow are encrypted and integrity protected.

5. The FAP initiates the IKE_AUTH exchange with the SeGW by setting the IDi payload to FEID in FQDN format (i.e., the value from the subjectAltName extension of the FAP certificate), CERT payload set to the FAP certificate corresponding to the FEID and the AUTH payload set according to section 2.15 of [6] using the private key corresponding to the FAP certificate. The authentication algorithm used to generate the AUTH payload is also included in the AUTH payload. The FAP also includes the CERTREQ payload containing the list of CA certificates for SeGW (server) authentication. For successful SeGW authentication, the CERTREQ payload has to contain at least one CA certificate that is in the trust chain of the SeGW certificate.

6. Using the CA certificate corresponding to the FAP certificate, the SeGW first verifies that the FAP certificate in the CERT payload has not been modified and the FAP identity included in the IDi payload corresponds to the identity in the FAP certificate. If the verification is successful, using the public key of the FAP certificate, the SeGW generates the expected AUTH payload and compares it with the received AUTH payload. If they match, then the authentication of the FAP is successful. Otherwise, the SeGW sends an IKEv2 Notification message indicating authentication failure.

7. If the network policy requires femtocell subscription authorization, the SeGW contacts the Femtocell AAA to verify that the FAP identified by FEID is authorized to provide service.

8. Femtocell AAA responds with the authorization result. If the authorization is not successful, the SeGW sends an IKEv2 Notification message indicating authorization failure. Otherwise, the SeGW proceeds with server authentication.

9. SeGW responds with the IKE_AUTH Response by setting the IDr payload to the FQDN (or IP address) of the SeGW present in the subjectAltName of the SeGW certificate, CERT payload set to the SeGW certificate corresponding to the FQDN (or IP address) and the AUTH payload set according to section 2.15 of [6] using the private key corresponding to the SeGW certificate. The authentication algorithm used to generate AUTH payload is also included in the AUTH payload.

10. Using the CA certificate corresponding to the SeGW certificate, the FAP first verifies that the SeGW certificate in the CERT payload has not been modified and the identity included in the IDr payload corresponds to identity in the server certificate and contains the expected SeGW value as discovered during the SeGW discovery procedures. If the verification is successful, using the public key of the FAP server certificate, the FAP generates the expected AUTH payload and compares it with the received AUTH payload. If they match, then the SeGW (server) authentication is successful. Otherwise, the FAP sends an IKEv2 Notification message indicating authentication failure. This completes the IKE_AUTH exchange.

11. An IPsec SA pair is established between the FAP and the SeGW. If more IPsec SA pair(s) are needed, either the FAP or the SeGW may initiate creation of CHILD_SA pairs using CREATE_CHILD_SA exchange.

12. The FAP proceeds to securely perform the rest of initialization procedures, such as FAP location determination, FAP configuration download, etc., with the cdma2000 core network as specified in [2].

7.3 FAP Authorization Mechanisms

The SeGW shall be capable of checking the network policy regarding whether a particular FAP is authorized to provide service. For example, if the network policy states that all FAPs that pass device authentication are authorized to provide service, then no further authorization check may be necessary. However, if the network policy requires that each FAP be individually authorized for service (e.g., the FEID is associated with a valid subscription, see steps 7 to 9 of figure 2), then the SeGW shall be capable of sending the AAA access request message to the Femtocell AAA using the protocols specified in [2]. If the Femtocell AAA responds affirmatively, then the SeGW shall proceed with the device authentication. Otherwise, the tunnel setup shall be terminated by sending an IKEv2 Notification message indication authentication failure.

7.4 Integrity Protection Mechanisms

The integrity of the IP packets sent through the tunnel between the FAP and the SeGW shall be protected using IPsec ESP [7].

7.5 Confidentiality Protection Mechanisms

The confidentiality of the IP packets sent through the tunnel between the FAP and the SeGW, if required, shall be protected using IPsec ESP [7].

7.6 Profile for FAP Certificates

The FAP certificate is used by the SeGW to authenticate the FAP. The FAP certificate shall be issued by the FAP vendor during its manufacturing. The private key associated with the FAP certificate shall be stored securely inside the FAP.

The FAP certificate profile shall be compliant to [11] and certificate profiles specified in [12] for IKEv2. In addition, the FAP certificate shall meet the profile defined below:

1. The signature algorithm used by the CA to sign the certificate shall be "sha256WithRSAEncryption", and the RSA public key used for signing shall be at least 2048 bits and shall be based on PKCS #1 version 2.1 defined in [13]. The algorithm identifier for "sha256WithRSAEncryption" is defined in [14] .
2. The issuer name shall not be empty and shall identify the name of the issuer as defined in [11] section 4.1.2.4.
3. The subject public key shall use algorithm "rsaEncryption" [14], and the RSA public key value shall be at least 2048 bits.

4. The subjectAltName extension shall be present for FAP certificate and shall contain FEID conforming to IEEE EUI-64 format identifying the IEEE hardware address of the FAP. The FEID, represented as a string of hexadecimal digits including any leading zeros, shall be encoded in the subjectAltName extension in FQDN format with FEID as the first label (e.g., FEID.vendor.com).

   NOTE: The FEID only needs to be encoded in FQDN format in subjectAltName extension and does not have to map to any real IP address.

5. The FAP certificate shall contain validity time and the Validity encoding shall be as specified in [11].

An example FAP certificate that satisfies the profile defined in this section is given in Annex A.

7.6.1 SeGW/IKEv2 Processing Requirements for FAP Certificates

The SeGW/IKEv2 processing requirements for FAP certificates are defined below:

1. The processing of the FAP certificate by SeGW shall be compliant to [12].

2. The FAP shall not send certificate paths containing more than four certificates.

3. The SeGW shall be able to support FAP certificate paths containing up to four certificates. The intermediate (FAP) CA certificates and the FAP certificate are obtained from the IKEv2 CERT payload and the trusted root CA or trusted intermediate CA certificate is obtained from a SeGW local store of trusted CA certificates.

4. The SeGW shall check the validity time of the FAP certificates, and reject certificates that are either not yet valid or that are expired.

7.7 Profile for SeGW Certificates

The SeGW certificate is used by the FAP to authenticate the SeGW. The SeGW certificate shall be issued to the SeGW using an operator trusted CA. The private key associated with the SeGW certificate shall be stored securely inside the SeGW.

The SeGW certificate profile shall be compliant to [11] and certificate profiles specified in [12] for IKEv2. In addition, the SeGW certificate shall meet the profile defined below:

1. The signature algorithm used by the CA to sign the certificate shall be "sha256WithRSAEncryption", and the RSA public key used for signing shall be at least 2048 bits and shall be based on PKCS #1 version 2.1 defined in [13]. The algorithm identifier for "sha256WithRSAEncryption" is defined in [14].

2. The issuer name shall not be empty and shall identify the name of the issuer as defined in [11] section 4.1.2.4.

3. The subject name may be empty in SeGW certificates and shall not be empty in CA certificates

4. The subject public key shall use algorithm “rsaEncryption” [14], and the RSA public key value shall be at least 2048 bits.

5. The subjectAltName extension shall be present if this is a SeGW certificate, and shall contain FQDN (if DNS is available) or IP address (if DNS is not available). However, the use of FQDN is strongly recommended.

6. The SeGW certificate shall contain validity time and the Validity encoding shall be as specified in [11].

An example SeGW certificate that satisfies the profile defined in this section is given in Annex A.

7.7.1 FAP/IKEv2 Processing Requirements for SeGW Certificates

The FAP/IKEv2 processing requirements for SeGW certificates are defined below:

1. The processing of the SeGW certificate by FAP shall be compliant to [12].

2. The SeGW shall not send certificate paths containing more than four certificates.

3. The FAP shall be able to support SeGW certificate paths containing up to four certificates. The intermediate (SeGW) CA certificates and the SeGW certificate are obtained from the IKEv2 CERT payload and the trusted root CA or trusted intermediate CA certificate is obtained from a FAP local store of trusted CA certificates.

4. The FAP shall check the validity time of the SeGW certificates, and reject certificates that are either not yet valid or that are expired. Optionally, the FAP may check the revocation status of the SeGW certificates.

7.8 Profile of IKEv2

The following IKEv2 profile shall be supported by the FAP and the SeGW:

IKE_SA_INIT exchange:

- Confidentiality: AES with 128-bit keys in CBC mode;
- Pseudo-Random Function: AES-XCBC-PRF-128;
- Integrity: HMAC-SHA1-96; support and use of AES-XCBC-MAC-96 is recommended;
- Diffie -Hellman group 2 1024-bit MODP; support and use of Diffie-Hellman Group 14 2048-bit MODP is recommended.

For NAT traversal, the NAT support of IKEv2 shall be supported as specified in section 2.23 of [6]. Re-keying of IPsec SAs and IKE SAs shall be supported as specified in [6].

IKE_AUTH exchange:

- The use of FAP (device) certificate for authentication of the FAP shall be supported. A profile for the FAP certificate is defined in section 7.5;
- The use of server certificate for SeGW authentication to the FAP shall be supported. A profile for the SeGW certificate is defined in section 7.6;
- Femtocell Equipment Identifier (FEID) encoded in FQDN format for Femtocell APs and FQDN for SeGWs shall be supported for identification.

CREATE_CHILD_SA exchange:

- Perfect Forward Secrecy is optional.

7.9 Profile of IPSec

The following IPSec profile shall be supported by the FAP and the SeGW:

- Protocol: ESP
- Confidentiality: AES with 128-bit keys in CBC mode;
- Integrity: HMAC-SHA1-96; support and use of AES-XCBC-MAC-96 is recommended;
- Tunnel mode shall be used.

It shall be possible to negotiate confidentiality protection with the transform IDs for encryption ENCR_NULL as specified in [6]. Integrity protection shall always be used and the authentication algorithm shall not be NULL.

For NAT traversal, the UDP encapsulation for ESP tunnel mode as specified in [8] shall be supported.

7.10 FAP - FMS Security

The FAP shall support TLS for secure communication with the FMS. The FAP shall be able to authenticate the FMS based on FMS's certificate. The FAP shall be authenticated by the FMS using the FAP certificate.

The FMS server shall support TLS to establish secure connection with the FAP. The FMS shall support certificate based authentication of the FAP using the FAP certificate and authenticate itself to the FAP using its server certificate (signed by an operator trusted CA).

The FAP and the FMS shall support TLS v1.1 [17] and may support TLS v1.2 [18].

For the TLS connection between the FAP and the FMS, the FAP and the FMS shall support the TLS cipher suite TLS_RSA_WITH_AES_128_CBC_SHA [19].

When the FMS is placed inside the operator's core network, the FAP communicates with the FMS via the SeGW. In this case, TLS may be used for further protection.

When the FMS is in the public domain, all traffic between the FAP and the FMS shall be protected using a mutually authenticated TLS connection. The mutual authentication shall be performed using the FAP and the FMS certificates.

7.10.1 Signed File Transfer

The file transfer operations between the FAP and the FMS (or another server) shall be signed using the private key corresponding to a certificate that is verifiable by the FAP. The file transfer shall use the Signed Package Format as specified by Broadband Forum TR-069 Amendment 2 [16]. When using the Signed Package Format to transfer files from FMS (or another server) to the FAP, the following requirements shall be met:

- The signature field in the Signed Package Format shall contain at least one signature signed by an entity trusted by the FAP (i.e., the CA certificate used for signing is in the trusted CA store of the FAP), together with a certificate or a certificate chain that can be verified by the FAP.
- The FAP shall verify both the certificate(s) and the signature of the downloaded file before taking any action (e.g. extraction, execution, etc.) on the file.
- If signature verification fails, the FAP shall discard the downloaded file and report to FMS.

8 cdma 2000 1x Femtocell System Specific Procedures

8.1 1x FAP IMS Security

For providing 1x Circuit Switched services, the FAP registers to the operator's network using IMS-based procedures as specified in [3]. The FAP registers to an entity called FCS (Femtocell Convergence Server) using IMS registration procedures. Unless configured otherwise, the FAP uses the femtocell identity to derive the IMS identities required to register with the FCS.

The FAP's SIP registration to the S-CSCF in the IMS domain shall be authenticated. The FAP shall use one of the following security mechanisms for registration to S-CSCF: Trusted Node Authentication (TNA), or SIP Digest and shall be compliant to [10]. TNA shall be used as the default authentication method unless the FAP is configured to use SIP Digest. If the FAP is configured to use SIP Digest, then the credentials required for these methods shall be configured and stored securely by the FAP. How the credentials for SIP Digest are configured is outside the scope of this document.

In the Trusted Node Authentication method, access to IMS is granted based on a successful access level authentication performed by a trusted node in the network. In the femtocell systems security architecture, the SeGW authenticates the FAP before access to the network is granted. When the TNA method is used, it relies on the fact that the FAP is authenticated by a SeGW at the home system before access to the core network is granted. Therefore, in the TNA method, the FAP acts as a trusted node to the IMS domain and takes on the role of both the SIP User Agent and the P-CSCF from an IMS authentication perspective [10], by setting the "integrity protected" flag to "auth-done" in the authorization header as specified in Annex P of [10].

8.2 RAND Generation by 1x FAP

This section specifies how the authentication global random challenge value (RAND) is generated by the FAP for authentication of the 1x AT using CAVE algorithm for accessing 1x services [9].

In order to enforce the expected voice privacy level, the FAP shall not be allowed to generate Global RAND values without the core network assistance for 1x authentication. Therefore, the core network based Global RAND generation mechanism specified in this section shall be used.

8.2.1 Core Network Based Global RAND generation

During the FAP's registration to FCS using SIP, the FCS shall assign a randomly selected 64-bit key (called GLOBAL_RAND_KEY) to the FAP registration session and send it to the FAP using a SIP message. The FCS shall not share the same GLOBAL_RAND_KEY with multiple FAPs.

After receiving the GLOBAL_RAND_KEY, the FAP may derive global RAND as needed (e.g. for broadcast in 1x OMT message).

The global RAND derivation shall be as follows:

RAND = 32 Least Significant Bits of KDF (GLOBAL_RAND_KEY, NONCE)

Where, the NONCE is a 32-bit unsigned integer and shall be an always increasing value for a given GLOBAL_RAND_KEY (i.e., value greater than any previously used value for a given GLOBAL_RAND_KEY, e.g., current system time in seconds) and the KDF shall be HMAC-SHA256 as per [20].

Upon receiving AUTHR from the AT in a 1x signaling message (e.g., System Access message), the FAP shall send the message, along with RAND/AUTHR pair, and the NONCE value used to derive the RAND, to the FCS encapsulated in a SIP message [3]. If the FCS has a stored NONCE value associated with the FAP (e.g., as part of the FAP session), then it shall verify that the received NONCE value is greater than the stored value. If not, the FCS shall reject the SIP message. Otherwise, the FCS shall compute the expected global RAND value using the received NONCE and the stored GLOBAL_RAND_KEY (as defined by the KDF above) to ensure that the received RAND value is fresh. If the received RAND matches the global RAND computed by the FCS, the FCS shall store the received NONCE and then proceed to contact the appropriate cdma2000 1x core network entity to continue with the rest of the 1x procedures.

The stage 3 details and call-flows for 1x FAP using IMS procedures are defined in [3].

9 HRPD and 1x Packet Data Femtocell System Specific Procedures

The FAP may provide packet data services using cdma2000 air interfaces (e.g., HRPD, 1x Packet Data). The stage 2/3 details of the interfaces between the FAP and the other components of the Radio Access Network (RAN) are specified in [5].

Annex A (Informative): Example Certificates

This annex provides example X.509 certificates that meet the certificate profile requirements defined in the main body of this document. The certificates have been decoded into a human readable form.

The following figure illustrates the certificate hierarchy of the example certificates.

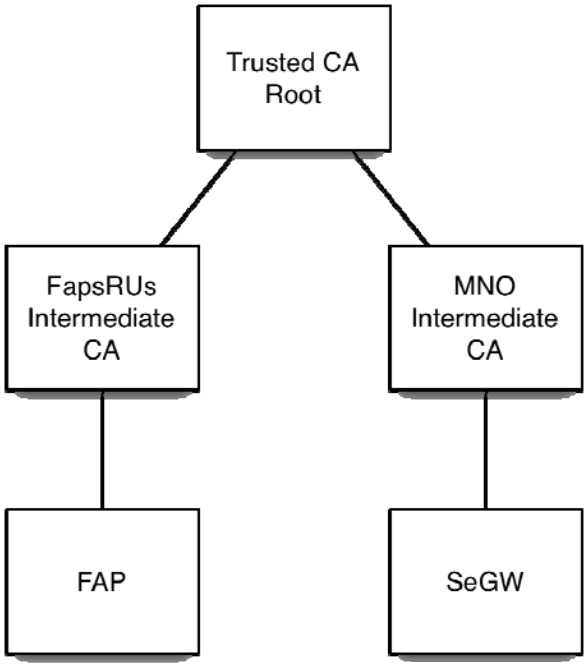

Figure A.1 Certificate Hierarchy

NOTE: In this annex, "FapsRUs" refers to a name of a fictitious FAP manufacturer.

In the following section for example certificates, the first column is the offset from the beginning of the certificate and the second column is the length (in octets) of the TLV structure.

A.1 Example FAP Certificate

The following is an example of a FAP certificate with FEID hexadecimal value of "*baadfeedbaadfeed*", which has been signed by the Intermediate CA certificate:

```
   0  795: SEQUENCE {
   4  515:   SEQUENCE {             -- TBSCertificate
   8    3:     [0] {
  10    1:       INTEGER 2          -- Version v3
        :       }
  13    4:     INTEGER 1247776555   -- CertificateSerialNumber
  19   13:     SEQUENCE {
  21    9:       OBJECT IDENTIFIER  -- AlgorithmIdentifier
        :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
  32    0:       NULL
```

```
 1             :       }
 2      34    44:     SEQUENCE {                  -- Issuer
 3      36    16:       SET {
 4      38    14:         SEQUENCE {
 5      40     3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
 6      45     7:           PrintableString 'FapsRUs'
 7             :           }
 8             :         }
 9      54    24:       SET {
10      56    22:         SEQUENCE {
11      58     3:           OBJECT IDENTIFIER commonName (2 5 4 3)
12      63    15:           PrintableString 'Intermediate CA'
13             :           }
14             :         }
15             :       }
16      80    30:     SEQUENCE {                  -- Validity dates
17      82    13:       UTCTime 16/07/2009 20:36:02 GMT
18      97    13:       UTCTime 15/07/2013 20:36:02 GMT
19             :       }
20     112    45:     SEQUENCE {                  -- Subject
21     114    16:       SET {
22     116    14:         SEQUENCE {
23     118     3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
24     123     7:           PrintableString 'FapsRUs'
25             :           }
26             :         }
27     132    25:       SET {
28     134    23:         SEQUENCE {
29     136     3:           OBJECT IDENTIFIER commonName (2 5 4 3)
30     141    16:           PrintableString 'baadfeedbaadfeed'
31             :           }
32             :         }
33             :       }
34     159   288:     SEQUENCE {                  -- SubjectPublicKeyInfo
35     163    13:       SEQUENCE {
36     165     9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
37     176     0:         NULL
38             :         }
39     178   269:       BIT STRING, encapsulates {
40     183   264:         SEQUENCE {
41     187   257:           INTEGER         -- Public modulus
42             :             00 BE 53 F2 18 22 82 09 FB 52 89 B5 43 E2 80 08
43             :             3F 65 88 19 57 9A F5 69 C2 43 72 C4 A8 04 2B E8
44             :             B7 0D 44 31 F8 AB 74 8B B4 FB 65 96 9A 48 99 95
45             :             F4 60 51 B2 A5 15 8E 6B 44 F0 7B 25 0D 22 3A A3
46             :             59 44 59 8C EE 00 18 AC 27 14 95 FF A4 ED A4 77
47             :             4B 44 C4 8B 17 1D 69 96 7A 91 21 92 E7 1B 98 0D
48             :             00 A9 A4 D4 51 48 43 C7 FE 71 F5 4C 9D 5B CF B1
49             :             1C CB DC 48 66 7A 56 26 FA 0A 95 F6 2A 5F 8B 15
50             :             2D CB 6F 39 01 11 33 F7 55 37 5D 2F 4B 08 CC 91
51             :             F4 B5 AC FE 56 E3 80 68 2F 20 B9 F8 B3 D2 2B 1D
52             :             CB 83 83 43 36 72 32 BE 6F 75 DA 0D 94 5A 4D 7A
53             :             68 44 13 2F A3 0A C0 E8 60 3D 05 DA E2 25 22 2E
54             :             76 42 FA C6 46 53 F4 EA AA 67 AB E2 D9 DA 3D E6
55             :             D4 01 62 B1 43 21 3A 3D E8 FE 20 26 CA A8 6F 44
56             :             EE A6 25 B7 EF 48 D1 7E DA 7D A9 DE 96 92 8C C7
57             :             2B FD 87 CC 5F 7D D0 EF 55 6E EF 9A 09 28 94 3D
58             :             5F
59     448     1:           INTEGER 3           -- Public exponent
60             :           }
61             :         }
62             :       }
63     451    70:     [3] {                       -- Extensions
64     453    68:       SEQUENCE {
```

```
  455    9:          SEQUENCE {
  457    3:            OBJECT IDENTIFIER basicConstraints (2 5 29 19)
  462    2:            OCTET STRING, encapsulates {
  464    0:              SEQUENCE {}  -- Is not a CA certificate
         :              }
         :            }
  466   14:          SEQUENCE {
  468    3:            OBJECT IDENTIFIER keyUsage (2 5 29 15)
  473    1:            BOOLEAN TRUE
  476    4:            OCTET STRING, encapsulates {
  478    2:              BIT STRING 7 unused bits
         :                '1'B (bit 0)
         :              }
         :            }
  482   39:          SEQUENCE {
  484    3:            OBJECT IDENTIFIER subjectAltName (2 5 29 17)
  489   32:            OCTET STRING, encapsulates {
  491   30:              SEQUENCE {
  493   28:                [2] 'baadfeedbaadfeed.fapsrus.com'
         :                }
         :              }
         :            }
         :          }
         :        }
         :      }
  523   13:    SEQUENCE {
  525    9:      OBJECT IDENTIFIER          -- AlgorithmIdentifier
         :        sha256withRSAEncryption (1 2 840 113549 1 1 11)
  536    0:      NULL
         :      }
  538  257:    BIT STRING             -- signatureValue
         :      89 4A F1 CB EE A9 49 70 4C AD 39 72 E4 CE 35 FF
         :      F8 43 DF C5 0A D7 FB D1 F0 83 CA 76 D1 16 8F F4
         :      54 14 14 46 20 8C D9 B5 A7 31 15 E4 46 D4 73 7D
         :      0D E7 9C 5A 6A 8A 5F 6D D4 1E 65 7C B9 49 C0 4D
         :      EE A4 B0 70 BA 56 D3 BB D9 34 29 34 AD 8F 7B 32
         :      4C C0 68 58 A9 0B E6 06 CE EC 49 B1 DA E3 5D C2
         :      F1 80 0E 63 92 87 B9 7D D7 E3 A2 89 E4 13 B8 3B
         :      37 3D DD FF BA 7C BD 45 9E 06 F4 E6 E6 CC 14 63
         :      A5 6C 66 E2 AC 26 EC 5E CE 01 81 34 D5 82 40 4F
         :      A9 E6 4D 28 DA 9F 51 45 06 97 41 74 34 C2 E5 24
         :      48 76 4E AF 2A 23 33 54 26 0B C7 E5 38 E1 03 DC
         :      B2 D0 6F A8 B5 78 7B C6 94 63 0B 8B 26 6C 10 8D
         :      D8 23 28 82 AE 39 5C A5 B9 BB C1 FD AF 88 4D 4E
         :      BF 74 62 B3 4A 2D C6 32 FA 9C FD A5 D7 9A 12 59
         :      B5 7B 90 BC B4 B2 AA D1 47 98 73 BF 94 D7 FC 6E
         :      70 33 E0 1B A5 26 B5 3C 0A CD 4B 01 7F 81 5B 4E
         :    }
```

A.2 Example FAP Intermediate CA Certificate

The following is an example of a FAP Intermediate CA certificate, which has been signed by a "Trusted CA" root certificate:

```
    0  879: SEQUENCE {
    4  471:   SEQUENCE {             -- TBSCertificate
    8    3:     [0] {
   10    1:       INTEGER 2          -- Version v3
         :       }
   13    4:     INTEGER 1247776554   -- CertificateSerialNumber
```

```
 1     19   13:     SEQUENCE {
 2     21    9:       OBJECT IDENTIFIER
 3              :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
 4     32    0:       NULL
 5              :       }
 6     34   36:     SEQUENCE {                  -- Issuer
 7     36   19:       SET {
 8     38   17:         SEQUENCE {
 9     40    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
10     45   10:           PrintableString 'Trusted CA'
11              :           }
12              :         }
13     57   13:       SET {
14     59   11:         SEQUENCE {
15     61    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
16     66    4:           PrintableString 'Root'
17              :           }
18              :         }
19              :       }
20     72   30:     SEQUENCE {                  -- Validity dates
21     74   13:       UTCTime 16/07/2009 20:35:59 GMT
22     89   13:       UTCTime 14/07/2017 20:35:59 GMT
23              :       }
24    104   44:     SEQUENCE {                  -- Subject
25    106   16:       SET {
26    108   14:         SEQUENCE {
27    110    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
28    115    7:           PrintableString 'FapsRUs'
29              :           }
30              :         }
31    124   24:       SET {
32    126   22:         SEQUENCE {
33    128    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
34    133   15:           PrintableString 'Intermediate CA'
35              :           }
36              :         }
37              :       }
38    150  288:     SEQUENCE {                  -- SubjectPublicKeyInfo
39    154   13:       SEQUENCE {
40    156    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
41    167    0:         NULL
42              :         }
43    169  269:       BIT STRING, encapsulates {
44    174  264:         SEQUENCE {
45    178  257:           INTEGER         -- Public modulus
46              :             00 E7 B5 6F 1B 8B 8E 5E C7 DA 6B 6C 12 85 57 2A
47              :             37 45 C6 CC 32 1D DE 3A 65 3F CE 2F 11 49 A0 36
48              :             C9 13 3D CF 07 2E FB 7D 89 38 9D DE 3E 2D 27 C2
49              :             06 8A 86 81 9F 93 BD 46 A1 D1 DC A2 B9 62 46 19
50              :             87 31 70 90 B4 F5 16 1C E8 24 9A E7 E9 19 C6 8F
51              :             E9 C6 FE 34 75 E7 D5 EE B8 53 F9 CA E3 D1 13 1F
52              :             F2 48 A9 80 AE 6A E3 D3 D3 DE AC CF 92 E1 0A DA
53              :             F0 E9 5C 58 9B 9A BD B9 DA 0E 68 08 EC 68 EA 37
54              :             15 CC 7D 00 4D F0 04 01 25 19 46 B2 A3 21 F4 11
55              :             63 BC D7 35 DE 23 33 3A 75 E0 73 71 B6 3B 6B 55
56              :             DD 92 39 6D 28 53 19 14 C5 1B DC E0 1D D3 7D 0E
57              :             50 DA CC CF 16 C9 86 B8 29 FB 35 CE 02 BD B0 21
58              :             21 F8 60 41 8A 28 4F 2B 69 7E 3E F1 E2 F3 8B 8A
59              :             B3 4F 44 1A 29 45 6B 72 1C 30 C2 16 B5 A0 DE 0F
60              :             D3 AC A1 E3 30 C7 84 D8 84 8E B5 03 20 0E 72 72
61              :             BC 52 56 19 DD 93 BE B1 DF 55 CC E5 60 46 3D DD
62              :             07
63    439    1:           INTEGER 3            -- Public exponent
64              :           }
```

```
          :         }
          :       }
  442   35:     [3] {                       -- Extensions
  444   33:       SEQUENCE {
  446   15:         SEQUENCE {
  448    3:           OBJECT IDENTIFIER basicConstraints (2 5 29 19)
  453    1:           BOOLEAN TRUE
  456    5:           OCTET STRING, encapsulates {
  458    3:             SEQUENCE {
  460    1:               BOOLEAN TRUE     -- Is a CA certificate
          :               }
          :             }
          :           }
  463   14:         SEQUENCE {
  465    3:           OBJECT IDENTIFIER keyUsage (2 5 29 15)
  470    1:           BOOLEAN TRUE
  473    4:           OCTET STRING, encapsulates {
  475    2:             BIT STRING 2 unused bits
          :               '100000'B (bit 5)
          :             }
          :           }
          :         }
          :       }
          :     }
  479   13:   SEQUENCE {
  481    9:     OBJECT IDENTIFIER          -- AlgorithmIdentifier
          :       sha256withRSAEncryption (1 2 840 113549 1 1 11)
  492    0:     NULL
          :     }
  494  385:   BIT STRING             -- signatureValue
          :     20 AE 3A E4 98 C1 64 64 9A DA C2 4C 89 3F 11 E8
          :     08 06 A2 0B 17 62 9E 50 DB A3 62 2D ED 4C C7 3F
          :     F0 AB 00 F3 5D 88 CC 9F 91 76 98 AB 26 8F 5E 0F
          :     6E 3F DF F1 B9 98 C6 F0 B4 FC 96 87 CC 5E 2F 9C
          :     C2 01 72 FE B3 B2 9A E2 C3 02 69 1B 06 09 AC D0
          :     68 F0 3B 1B 61 F1 C9 CD 7D 55 E7 0B 41 65 8A 86
          :     27 E6 C5 C4 51 C5 F7 05 25 5D 30 A4 93 77 EE 7E
          :     CC F2 26 4D BF 75 81 A6 8D 96 DB E3 DF 9B F7 E8
          :     90 68 36 06 11 D1 16 0B 95 94 14 09 F2 6C EA 31
          :     48 3B 80 A4 8A 2B 0C BF 92 D0 CA 1B C4 85 B4 51
          :     A1 10 6D C0 8C CA 8B 28 97 A9 48 E3 21 EF CF 25
          :     00 35 77 AA 07 09 82 57 47 F3 32 20 E0 D4 B6 AB
          :     81 D8 42 15 80 06 10 F6 6F 84 55 17 34 53 98 2C
          :     DC E1 C8 AE 54 51 41 AD B1 3F 91 E5 6E E6 10 4F
          :     B2 C2 84 A4 43 A3 2B E2 87 AA 18 9E 2A 01 C1 46
          :     CA 36 40 A4 E7 A0 99 FB 4E FB 7C 7C DB 61 26 5C
          :     FD 9B 19 C8 9E D7 EC 10 FE 53 6B 90 C7 C7 BB EA
          :     B5 3D 06 FA 69 B8 60 23 26 20 21 3A 89 15 FB 32
          :     6E 9A EC AB D7 41 FC 29 16 64 0A 7C 7F BA 7F 6E
          :     AB 87 71 F7 16 EF 94 61 E3 B6 F8 4A 05 12 18 EF
          :     3D 6D 7B D7 2C 1F 2D 91 17 DF 21 75 C1 11 BE B1
          :     7A 40 CE 0D ED AA 16 A2 B8 03 EF 5F 5D 7D B9 82
          :     5E 50 2D 39 9F CE 28 90 9F 76 7B DE EC 81 78 DC
          :     D2 91 FD 09 F1 ED D2 09 09 49 6F 66 EA DF F2 DE
          :   }
```

A.3 Example Root CA Certificate

For all the example certificates in this Annex, this certificate is used as the Root CA certificate. This Root CA is assumed to be trusted by both the FAP manufacturer and the Mobile Network Operator (MNO). Note that it is possible that the FAP manufacturer and/or the MNO are using a different CAs as their trusted CA and this case is not considered in the example certificates given in this Annex. The following is an example of a Root CA certificate for a trusted CA:

```
  0  999: SEQUENCE {
  4  591:   SEQUENCE {             -- TBSCertificate
  8    3:     [0] {
 10    1:       INTEGER 2                -- Version v3
       :       }
 13    4:     INTEGER 1247776553   -- CertificateSerialNumber
 19   13:     SEQUENCE {
 21    9:       OBJECT IDENTIFIER  -- AlgorithmIdentifier
       :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
 32    0:       NULL
       :       }
 34   36:     SEQUENCE {                 -- Issuer
 36   19:       SET {
 38   17:         SEQUENCE {
 40    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
 45   10:           PrintableString 'Trusted CA'
       :           }
       :         }
 57   13:       SET {
 59   11:         SEQUENCE {
 61    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
 66    4:           PrintableString 'Root'
       :           }
       :         }
       :       }
 72   30:     SEQUENCE {                 -- Validity dates
 74   13:       UTCTime 16/07/2009 20:35:56 GMT
 89   13:       UTCTime 12/07/2025 20:35:56 GMT
       :       }
104   36:     SEQUENCE {                 -- Subject
106   19:       SET {
108   17:         SEQUENCE {
110    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
115   10:           PrintableString 'Trusted CA'
       :           }
       :         }
127   13:       SET {
129   11:         SEQUENCE {
131    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
136    4:           PrintableString 'Root'
       :           }
       :         }
       :       }
142  416:     SEQUENCE {                 -- SubjectPublicKeyInfo
146   13:       SEQUENCE {
148    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
159    0:         NULL
       :         }
161  397:       BIT STRING, encapsulates {
166  392:         SEQUENCE {
170  385:           INTEGER        -- Public modulus
       :           00 B9 EC 94 CF 75 3E DD A3 79 FB 8C 07 5E 3A 89
       :           D7 B2 B1 10 14 F4 C2 B2 E0 12 B3 55 61 A9 D0 D5
```

```
  1            :           AD 5F 10 75 0C AD 60 6B 75 8F 43 16 EE DB 67 28
  2            :           39 6B 8F 6E E0 67 89 0C 79 C7 A0 C1 2C 12 BD 80
  3            :           25 EC 43 CE 23 D5 B9 61 10 65 37 96 2B 64 45 D1
  4            :           48 D3 AA 53 36 F9 29 8D BF 0A 69 CE 6D 75 76 79
  5            :           39 5D 0E 53 99 29 98 DA FA 39 D4 B7 E5 93 E8 4C
  6            :           99 54 7F 71 E2 49 3D 24 B1 72 91 42 9C 2A FD DF
  7            :           CC 37 03 BD 3B 36 92 63 82 F3 61 3D 50 D5 41 02
  8            :           35 76 5D 4D 1E E5 60 18 A3 6F D2 26 D6 E2 8F 50
  9            :           15 9B C8 CB 64 C8 0B C2 6F 90 1E 0B A9 98 08 70
 10            :           F9 16 67 5D 14 E9 DB F8 5F D3 15 B1 6B C2 4D 61
 11            :           E0 A7 4C 37 8D 0A E2 20 D5 BA C0 8D 65 4C BE 90
 12            :           A2 90 D9 4C 8F EA E0 5D 1E 65 C0 FF 27 1B 35 A9
 13            :           70 52 58 E3 A4 35 0D 40 A1 A4 32 30 CD 8B 63 77
 14            :           C8 F9 35 54 C7 31 5E B0 78 9D F6 2D C9 C6 33 F4
 15            :           02 B0 F4 20 93 E4 B6 FF B3 40 B4 05 4C 38 9B B8
 16            :           24 3A AF 5B 49 96 FF 19 26 8F 76 76 FC 1B 7F EF
 17            :           1E 0E 08 96 69 83 F5 E8 F7 AD 04 CD 30 4E 5A 54
 18            :           91 62 FC 6F 2F 8F 26 C1 A7 13 9C C9 58 55 8C 53
 19            :           BA 03 4B 29 DC 32 F7 97 44 E6 4B D5 D8 C5 12 23
 20            :           67 AA A6 3D 27 55 63 88 A6 3C 4E 58 91 08 E6 07
 21            :           F6 58 E3 50 31 99 33 47 9A D4 0A 53 63 94 AD 82
 22            :           32 56 D3 95 FD A5 AD 02 13 E8 C3 7C 9C BD 98 BE
 23            :           5B
 24    559    1:           INTEGER 3             -- Public exponent
 25            :           }
 26            :         }
 27            :       }
 28    562   35:       [3] {                     -- Extensions
 29    564   33:         SEQUENCE {
 30    566   15:           SEQUENCE {
 31    568    3:             OBJECT IDENTIFIER basicConstraints (2 5 29 19)
 32    573    1:             BOOLEAN TRUE
 33    576    5:             OCTET STRING, encapsulates {
 34    578    3:               SEQUENCE {
 35    580    1:                 BOOLEAN TRUE     -- Is aCA certificate
 36            :                 }
 37            :               }
 38            :             }
 39    583   14:           SEQUENCE {
 40    585    3:             OBJECT IDENTIFIER keyUsage (2 5 29 15)
 41    590    1:             BOOLEAN TRUE
 42    593    4:             OCTET STRING, encapsulates {
 43    595    2:               BIT STRING 2 unused bits
 44            :                 '100000'B (bit 5)
 45            :               }
 46            :             }
 47            :         }
 48            :       }
 49            :     }
 50    599   13:   SEQUENCE {
 51    601    9:     OBJECT IDENTIFIER          -- AlgorithmIdentifier
 52            :       sha256withRSAEncryption (1 2 840 113549 1 1 11)
 53    612    0:     NULL
 54            :     }
 55    614  385:   BIT STRING             -- signatureValue
 56            :     60 FC 04 8B B7 CB 49 36 98 7E D0 56 05 51 E4 2C
 57            :     D6 02 FC DB A2 7D F1 F0 FE DC 64 73 38 75 AC DB
 58            :     D0 A4 F0 C3 3B 2B BF 7E 84 D9 B5 66 E8 4B C1 F2
 59            :     7A 11 38 C6 1D D3 58 B5 A0 EA 18 67 32 1E 83 F7
 60            :     5A 54 79 4A 68 6B 5A D3 28 21 F7 C6 C9 3B F2 B3
 61            :     F1 D6 FB F5 73 3B 1A 79 6B D9 D3 B6 9D E1 D3 AC
 62            :     94 2D F4 F3 C7 2E 50 6B 78 9D C5 95 E7 61 28 60
 63            :     08 85 38 1E C4 C3 37 10 74 80 C3 12 E4 C4 ED FA
 64            :     78 A9 6E 7D DC 4D C7 FC F2 9D 07 BA 97 B7 EA 60
```

```
          :     5F A9 3A 2D 63 36 A5 25 29 2F 16 9F B2 94 B0 1C
          :     E2 11 6E A3 95 FF D8 88 36 B9 7C 07 A5 F0 9E 5B
          :     29 CD B6 A6 2E 23 D9 DE A4 FC 9C 3A 18 D4 19 08
          :     91 60 A8 FF 2F BE F4 B6 E4 95 AE 94 9D DC 63 0E
          :     35 45 30 0A 41 DE E5 30 8A A0 45 E4 8D BB A1 8C
          :     6C 51 86 36 3C 4F 86 A1 12 FA E5 DB 0E 70 E9 A2
          :     2E 30 B7 0E 84 B9 0C 51 FE C8 20 BE F4 5C 9B 1B
          :     79 3D 84 75 BD D2 95 58 42 2E 16 46 C6 2B 0E 71
          :     29 B2 BF 67 7C E4 A6 67 56 1E D0 F6 20 6D EE 65
          :     39 DD F8 71 9B 6C 55 A6 D4 34 2F 90 5B 29 2A 8E
          :     D2 5D 92 37 48 D2 E3 0F 42 36 FA 23 1B AE 61 78
          :     10 7E 09 43 43 ED A3 4D D3 6D BF 25 25 8D 28 82
          :     CE 9D F3 B2 4C E6 D8 19 67 3D 2F 20 2E 2B 2C DE
          :     26 D9 7E BD A9 61 41 21 75 A5 00 ED 29 45 C1 F9
          :     BA 51 59 80 01 70 23 3E B1 A4 18 91 50 92 2D F2
          :   }
```

A.4 Example SeGW Certificate

The following is an example of a SeGW certificate:

```
    0  782: SEQUENCE {
    4  502:   SEQUENCE {             -- TBSCertificate
    8    3:     [0] {
   10    1:       INTEGER 2          -- Version v3
          :       }
   13    4:     INTEGER 1247776557   -- CertificateSerialNumber
   19   13:     SEQUENCE {
   21    9:       OBJECT IDENTIFIER  -- AlgorithmIdentifier
          :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
   32    0:       NULL
          :       }
   34   60:     SEQUENCE {                 -- Issuer
   36   32:       SET {
   38   30:         SEQUENCE {
   40    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
   45   23:           PrintableString 'Mobile Network Operator'
          :           }
          :         }
   70   24:       SET {
   72   22:         SEQUENCE {
   74    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
   79   15:           PrintableString 'Intermediate CA'
          :           }
          :         }
          :       }
   96   30:     SEQUENCE {                 -- Validity dates
   98   13:       UTCTime 16/07/2009 20:36:06 GMT
  113   13:       UTCTime 15/07/2013 20:36:06 GMT
          :       }
  128   29:     SEQUENCE {                 -- Subject
  130   12:       SET {
  132   10:         SEQUENCE {
  134    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
  139    3:           PrintableString 'MNO'
          :           }
          :         }
  144   13:       SET {
  146   11:         SEQUENCE {
```

```
 1    148    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
 2    153    4:           PrintableString 'SeGW'
 3           :           }
 4           :         }
 5           :       }
 6    159  288:     SEQUENCE {                  -- SubjectPublicKeyInfo
 7    163   13:       SEQUENCE {
 8    165    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
 9    176    0:         NULL
10           :         }
11    178  269:       BIT STRING, encapsulates {
12    183  264:         SEQUENCE {
13    187  257:           INTEGER        -- Public modulus
14           :           00 96 17 6A 99 31 D3 C7 DD 6F BB B1 41 E1 8C F1
15           :           0F B4 BF 62 F7 0D E4 E4 D2 33 74 B0 B1 EE FE 49
16           :           79 00 8E C6 1D F6 CF C0 E2 39 52 C3 CE D7 6E 5A
17           :           76 A1 3F 14 9C 39 62 9B BC C4 50 AC 36 AB 3A DF
18           :           C3 63 8B 03 38 19 0A 8F A5 EE 47 60 2F 40 0C F7
19           :           C5 7C 5E 16 61 49 4C 17 17 56 D5 99 6B 0F 4E FB
20           :           62 9A E6 54 84 68 1F 0A F9 97 F3 58 DF 19 C4 73
21           :           4D 18 CD B8 E1 17 96 16 0E 76 0F A5 57 37 51 F6
22           :           2B DF DC 79 D5 5A 84 E8 80 D8 0B CC BE 07 6A F2
23           :           6C CC 7E 59 A5 03 44 8C 54 A1 09 B8 75 C0 7D 29
24           :           E1 96 AA 56 C0 DC CA 3C FF 6C 59 0D 59 83 E6 36
25           :           38 EF 08 E5 29 82 1D 23 56 83 72 A6 AB B0 33 71
26           :           98 FB 2A 3B EC 30 52 F9 B8 5D 61 74 F5 73 DB CA
27           :           FE 02 81 72 39 FE A4 D8 47 4F 0F A6 92 00 14 E3
28           :           60 28 2C BF FE 6F A9 5B 55 7B 8D 84 AA 27 95 EA
29           :           1C 17 BD 4A F4 97 51 60 56 54 E7 B6 D3 0F B3 0B
30           :           23
31    448    1:           INTEGER 3            -- Public exponent
32           :           }
33           :         }
34           :       }
35    451   57:     [3] {                       -- Extensions
36    453   55:       SEQUENCE {
37    455   12:         SEQUENCE {
38    457    3:           OBJECT IDENTIFIER basicConstraints (2 5 29 19)
39    462    1:           BOOLEAN TRUE
40    465    2:           OCTET STRING, encapsulates {
41    467    0:             SEQUENCE {}  -- Is not a CA certificate
42           :             }
43           :           }
44    469   14:         SEQUENCE {
45    471    3:           OBJECT IDENTIFIER keyUsage (2 5 29 15)
46    476    1:           BOOLEAN TRUE
47    479    4:           OCTET STRING, encapsulates {
48    481    2:             BIT STRING 7 unused bits
49           :               '1'B (bit 0)
50           :             }
51           :           }
52    485   23:         SEQUENCE {
53    487    3:           OBJECT IDENTIFIER subjectAltName (2 5 29 17)
54    492   16:           OCTET STRING, encapsulates {
55    494   14:             SEQUENCE {
56    496   12:               [2] 'segw.mno.com'
57           :               }
58           :             }
59           :           }
60           :         }
61           :       }
62           :     }
63    510   13:   SEQUENCE {
64    512    9:     OBJECT IDENTIFIER          -- AlgorithmIdentifier
```

```
          :       sha256withRSAEncryption (1 2 840 113549 1 1 11)
  523     0:     NULL
          :     }
  525   257:   BIT STRING             -- signatureValue
          :     C1 8C A9 32 C5 BD 55 AC BD 7D 7E FB 10 9C 47 83
          :     A7 07 6A FF 5C FA EE E3 B3 35 B5 04 02 6B FF 95
          :     63 86 10 67 7D 82 A2 83 81 B4 74 42 69 72 7C 1B
          :     8C FB A3 AE DC A9 57 17 50 9B 63 9E 1E DA 79 F3
          :     51 B5 3F F1 40 6F F9 DB E9 30 4C ED 06 A0 15 A1
          :     30 4A AA 85 67 BF B6 4B D5 5D AF 85 3B EB 78 A7
          :     69 6D 64 4D 02 D0 9A 39 55 EA 06 63 2F 22 63 45
          :     CB 9F CE 26 33 84 E4 D6 6A FF 1B 4A BB 32 8C B3
          :     12 21 53 56 12 87 B8 B2 01 26 1B 86 46 B1 BB 63
          :     AB 29 96 C8 C3 14 93 D1 1F 58 F5 7F CD 08 30 9E
          :     C5 FE 09 F7 FD 58 0E D5 D0 FE 64 C8 2B 9D 66 56
          :     87 BF 9A 67 51 1E 1B D0 F2 88 47 93 CF E1 E7 2C
          :     E6 E0 C7 04 9A AF 80 BC 53 BD DE 8F AF A6 18 8D
          :     2D 05 A6 D6 55 39 AA 2C 42 2A CB 8D 50 F5 DF 95
          :     75 32 69 15 E7 51 78 60 62 C1 85 C1 69 9A 0B 36
          :     48 76 D5 52 7E 65 A8 6D 99 28 E7 42 8A DB C2 71
          :   }
```

A.5 Example MNO Intermediate CA Certificate

The following is an example of an intermediate CA certificate for an MNO (Mobile Network Operator), which has been signed by the “Trusted CA” root certificate:

```
    0   895: SEQUENCE {
    4   487:   SEQUENCE {             -- TBSCertificate
    8     3:     [0] {
   10     1:       INTEGER 2          -- Version v3
          :       }
   13     4:     INTEGER 1247776556   -- CertificateSerialNumber
   19    13:     SEQUENCE {
   21     9:       OBJECT IDENTIFIER
          :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
   32     0:       NULL
          :       }
   34    36:     SEQUENCE {                 -- Issuer
   36    19:       SET {
   38    17:         SEQUENCE {
   40     3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
   45    10:           PrintableString 'Trusted CA'
          :           }
          :         }
   57    13:       SET {
   59    11:         SEQUENCE {
   61     3:           OBJECT IDENTIFIER commonName (2 5 4 3)
   66     4:           PrintableString 'Root'
          :           }
          :         }
          :       }
   72    30:     SEQUENCE {                 -- Validity dates
   74    13:       UTCTime 16/07/2009 20:36:04 GMT
   89    13:       UTCTime 14/07/2017 20:36:04 GMT
          :       }
  104    60:     SEQUENCE {                 -- Subject
  106    32:       SET {
  108    30:         SEQUENCE {
  110     3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
```

```
    115   23:           PrintableString 'Mobile Network Operator'
      :                 }
      :               }
    140   24:         SET {
    142   22:           SEQUENCE {
    144    3:             OBJECT IDENTIFIER commonName (2 5 4 3)
    149   15:             PrintableString 'Intermediate CA'
      :                 }
      :               }
      :             }
    166  288:       SEQUENCE {                  -- SubjectPublicKeyInfo
    170   13:         SEQUENCE {
    172    9:           OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    183    0:           NULL
      :                 }
    185  269:         BIT STRING, encapsulates {
    190  264:           SEQUENCE {
    194  257:             INTEGER         -- Public modulus
      :                   00 E8 1F 30 02 E4 1F 25 1F CC 2D DA D3 C5 98 DA
      :                   40 A2 1F 55 0D F1 9D 18 E2 0B 3F 92 3F FB 16 98
      :                   25 B4 D1 39 DE 5A 30 E4 33 1E 1D 11 5E 01 19 7B
      :                   87 43 43 AE 1E 8A EE 58 2D CC 4A BB 9D 90 08 0D
      :                   F4 B0 45 D3 54 B4 21 6F A5 05 DF D1 BE 2B 63 15
      :                   DA 90 26 10 AA 80 B4 A8 FB 7F F8 B4 27 62 D1 2A
      :                   1C 62 49 B9 FA 35 A5 0D B0 CA FE 7E DB 89 8B 96
      :                   3F 49 91 C2 B2 D0 EF DD 47 46 81 9E 85 06 16 A3
      :                   8C 9A FC 2F 88 32 32 FF B3 2B 2C 3B EE F2 8C 2F
      :                   B1 20 37 C6 9A 5C C0 0B 72 73 0F D5 72 07 5B BB
      :                   AF B4 91 8F C5 D0 E3 15 6C 2B DF E8 46 F9 70 31
      :                   4A F5 65 00 1D 4A F6 67 8D D8 1C A7 59 DB 9C F7
      :                   21 54 31 55 99 2E DA 4C 11 BC 31 27 65 67 EB 3E
      :                   E7 90 0B 76 DA E1 CD BC 1D 10 88 39 2A 09 80 8A
      :                   D0 9E 6E 12 CB D1 0B 23 A8 E5 31 A2 76 71 31 61
      :                   64 8B 4C C2 B3 34 46 2C FC B0 D2 20 7D 61 28 DA
      :                   B5
    455    1:             INTEGER 3            -- Public exponent
      :                   }
      :               }
      :             }
    458   35:       [3] {                       -- Extensions
    460   33:         SEQUENCE {
    462   15:           SEQUENCE {
    464    3:             OBJECT IDENTIFIER basicConstraints (2 5 29 19)
    469    1:             BOOLEAN TRUE
    472    5:             OCTET STRING, encapsulates {
    474    3:               SEQUENCE {
    476    1:                 BOOLEAN TRUE     -- Is a CA certificate
      :                   }
      :                 }
      :               }
    479   14:           SEQUENCE {
    481    3:             OBJECT IDENTIFIER keyUsage (2 5 29 15)
    486    1:             BOOLEAN TRUE
    489    4:             OCTET STRING, encapsulates {
    491    2:               BIT STRING 2 unused bits
      :                   '100000'B (bit 5)
      :                 }
      :               }
      :             }
      :           }
      :       }
    495   13:   SEQUENCE {
    497    9:     OBJECT IDENTIFIER           -- AlgorithmIdentifier
      :         sha256withRSAEncryption (1 2 840 113549 1 1 11)
```

```
    508    0:      NULL
           :      }
    510  385:    BIT STRING             -- signatureValue
           :      41 6C 6F C9 F2 2E C3 E9 7E C5 61 9C 4B 9C 0F C6
           :      F6 85 FE F6 E5 A6 46 DC 85 E2 3D A5 86 0C 82 B1
           :      AC D8 8F 07 68 C3 FD 29 7F 80 FF 62 B4 46 84 2C
           :      94 F4 C4 15 F7 4D 19 BB D2 CC E8 B9 32 FD 24 74
           :      15 99 BC 7C 16 AA 4F 48 2F 7E 68 D3 27 50 8C 3D
           :      68 3F DD 9B 56 B2 A5 43 C8 3B B7 4A A4 9E E2 D6
           :      23 74 28 82 38 3D 38 D3 79 0B 0D 03 90 17 5F E3
           :      6D DA 26 0F EC 41 6F CB 82 1D 1F 57 02 7E 43 83
           :      6D 01 A8 78 3F 0F DB F5 C8 C0 73 C2 B2 3B 60 5E
           :      48 D8 6D 28 CA A4 B7 86 9B 52 17 B7 A1 E4 21 A3
           :      02 2D 02 1D E5 9F 66 EE 2B 81 D6 9A 87 10 E4 08
           :      75 15 1B 58 1A F8 82 4D 14 77 DF 59 51 61 14 A4
           :      25 99 53 85 BE 4E B9 19 B2 1E AA 92 5B DC 2A 6E
           :      AC EB DD 5C A1 FE 23 C6 E1 43 39 6B D6 A5 B2 65
           :      7F 98 7B 76 1C FC C5 B0 A5 D0 29 8E 5B EA 97 A6
           :      6B 25 3B 54 64 20 08 B7 DE D5 AF 11 53 18 AA F3
           :      4F 29 B3 CC 75 C8 C9 C3 1E D9 30 65 43 FA 33 23
           :      4A 07 EF 3E 4E 5B B0 0E B5 94 75 5E D9 F0 B5 55
           :      59 DA 03 E6 C7 C6 BB 94 B2 90 09 72 FE 48 56 4F
           :      1B 72 94 9B AF BE 37 48 6B 33 A9 53 05 7A E5 1E
           :      57 4F 3A 41 D0 57 14 AF 67 FE 97 F6 74 12 69 DD
           :      02 2E B3 2F CF 64 3E A2 4D 78 CB A7 73 9F 7D C4
           :      A9 A6 B0 49 B0 5F E9 F1 63 56 F5 60 7E 2D 63 6A
           :      29 DB D0 80 11 97 12 B9 6D E9 03 9C A5 AA C5 4D
           :    }
```

Annex B (Informative): Call Flows for legacy FAP authentication

This annex illustrates how legacy authentication methods, such as EAP-AKA <1>, used by legacy (i.e., pre-standards) FAPs, can co-exist with the certificate-based authentication method specified in the main body of this document for FAP device authentication. EAP-AKA requires that a shared root key (RK) of sufficient length (e.g., keys of length 128-bit or greater) and an associated FAP identity (e.g., FEID) are securely pre-configured on the FAP and shared between the FAP and the Femtocell AAA. How these credentials are configured is outside the scope of this document.

B.1 FAP authentication using EAP-AKA

This section illustrates the IKEv2 based call flow for authenticating a FAP using EAP-AKA <1>. The EAP-AKA authentication credentials, such as the AKA root key (RK), AKA Identity (e.g., FEID in NAI format) and the AKA algorithms, are assumed to be securely configured on the FAP and shared with the Femtocell AAA in the home network.

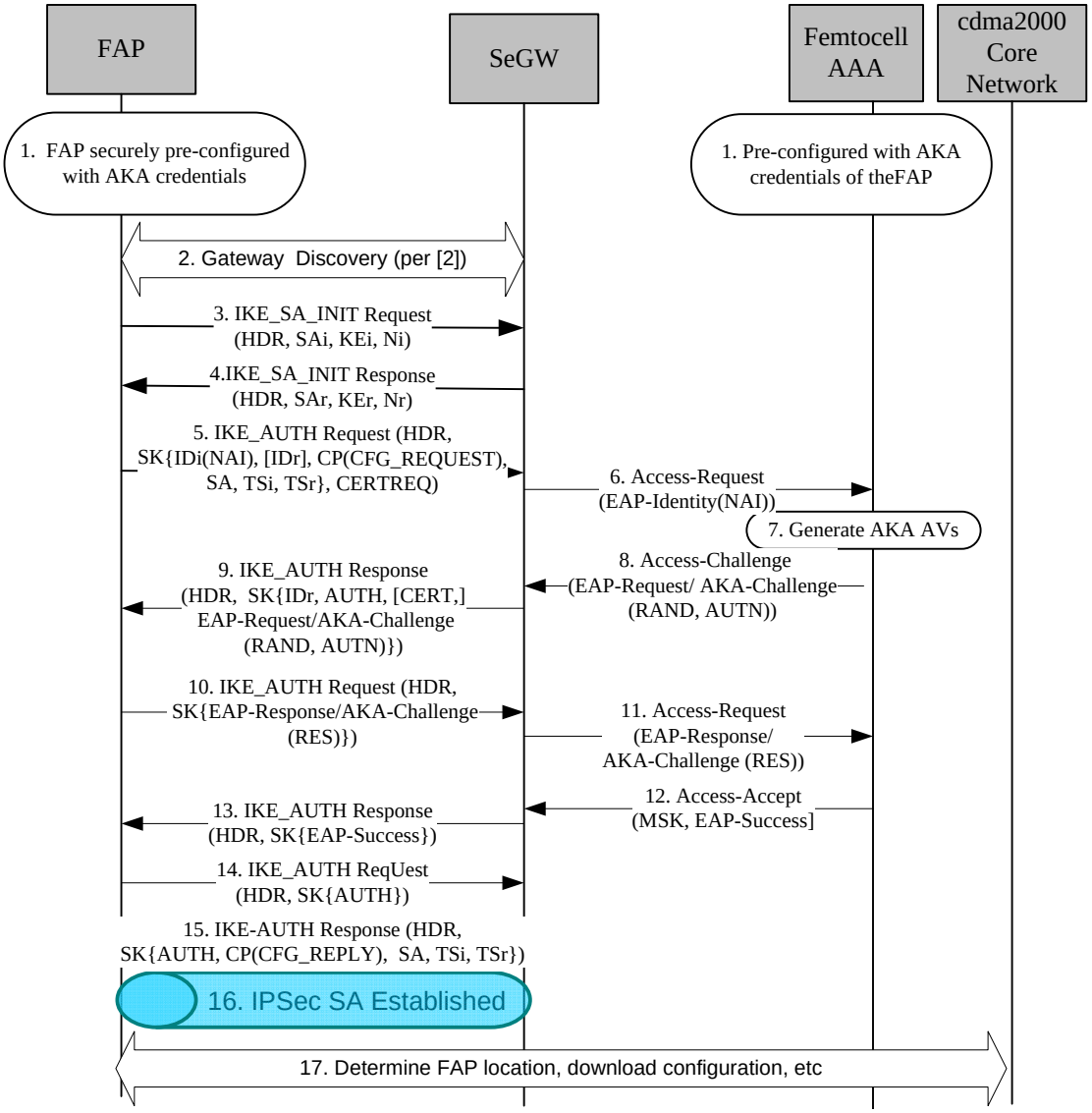

Figure B.1 FAP Authentication using EAP-AKA

1. The FAP and the AAA are preconfigured with the required AKA credentials (e.g., AKA root key, AKA identity, AKA algorithms etc.)
2. The FAP discovers the SeGW using procedures per [2].
3. The FAP sends an IKE_SA_INIT request to the SeGW.
4. The SeGW sends IKE_SA_INIT response. This completes the IKE INIT exchange.
5. The FAP starts the IKE AUTH exchange by sending an IKE_AUTH Request which includes the AKA identity of the FAP in NAI format (e.g., FEID@realm) in the IDi field. The AUTH payload is omitted to indicate that the FAP wants to use EAP for authentication. The FAP also requests a certificate from the SeGW. The FAP may also include a Configuration payload to assign an IP address dynamically.

6. The SeGW sends the Access-Request message to the AAA by including the FAP's EAP-AKA identity in NAI format (e.g., FEID@realm).

7. Based on the NAI, the AAA selects EAP-AKA as the authentication method and generates an AKA authentication vector (AV).

8. The AAA initiates the AKA-Challenge using the AKA AV.

9. The SeGW includes the AKA-Challenge in the IKE_AUTH response to the FAP. The SeGW identity, certificate and the AUTH parameter are also included. The AUTH parameter is used to protect the previous message that the SeGW send to the FAP (i.e., IKE_SA_INIT response in step 4).

10. The FAP executes AKA procedures and computes the response to the AKA-Challenge. The FAP also authenticates the SeGW based on the SeGW certificate.

11. The SeGW forwards the authentication response to the AAA.

12. If the authentication is successful, the AAA responds with an EAP-Success message including the MSK generated during the authentication process.

13. The EAP success message is forwarded to the FAP.

14. Using the MSK generated by the FAP during the AKA procedures, the FAP generates the AUTH parameter to authenticate the first IKE_SA_INIT message.

15. The SeGW uses the MSK received from the AAA to verify the AUTH parameter. If the verification is successful, the SeGW generates the AUTH parameter using the MSK for the second IKE_SA_INIT message and sends the IKE_AUTH message to the FAP. The SeGW includes the IP address assigned to the FAP in the configuration payload.

16. The FAP verifies the AUTH parameter. If successful, then the IPSec Security Association (SA) is established between the FAP and the SeGW. This completes the FAP authentication and IPSec tunnel establishment using IKEv2.

17. The FAP continues with the rest of the procedures as per [2].