

RSP Test Certificates Definition
Version 1.0
09 June 2017

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

Copyright Notice

Copyright © 2017 GSM Association

Disclaimer

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Antitrust Notice

The information contain herein is in full compliance with the GSM Association's antitrust compliance policy.

Table of Contents

1 Introduction 4
1.1 Scope 4
1.2 References 4
2 Tool chain for generation of the keys and certificates 4
2.1 OpenSSL 4
2.2 Keys generation 4
2.3 CI Certificate Generation 5
2.4 Non-Root Certificate generation 6
2.5 Certificate display 7
3 Test Certificates and keys 7
3.1 Certificate Issuer 7
3.1.1 CI Certificate: definition of data to be signed 8
3.1.2 CI Keys and Certificate 8
3.1.3 Input data for generation 9
3.2 eUICC 9
3.2.1 eUICC Certificate: definition of data to be signed 9
3.2.2 eUICC Keys and Certificate 10
3.2.3 Input data for generation 10
3.3 EUM 11
3.3.1 EUM Certificate: definition of data to be signed 11
3.3.2 EUM Keys and Certificate 12
3.3.3 Input data for generation 12
3.4 SM-DP+ 13
3.4.1 DPauth 13
3.4.1.1 SM-DP+ Certificate for Authentication: definition of data to be signed 13
3.4.1.2 SM-DP+ Keys and Certificate 14
3.4.1.3 Input data for generation 14
3.4.2 DPpb 15
3.4.2.1 SM-DP+ Certificate for Profile Binding: definition of data to be signed 15
3.4.2.2 SM-DP+ Keys and Certificate 16
3.4.2.3 Input data for generation 16
3.4.3 TLS 17
3.4.3.1 SM-DP+ TLS Certificate: definition of data to be signed 17
3.4.3.2 SM-DP+ TLS Keys and Certificate 18
3.4.3.3 Input data for generation 18
3.5 SM-DS 19
3.5.1 DSauth 19
3.5.1.1 SM-DS Certificate for Authentication: definition of data to be signed 19
3.5.1.2 SM-DS Keys and Certificate 19
3.5.1.3 Input data for generation 20
3.5.2 TLS 21
3.5.2.1 SM-DS TLS Certificate: definition of data to be signed 21
3.5.2.2 SM-DS TLS Keys and Certificate 21

3.5.2.3 Input data for generation 22
Annex A Document Management (Informative) 23
A.1 Document History 23

1 Introduction

1.1 Scope

This document's scope is to define the Test Certificates that will be used in the tests specified in SGP.23 [1] based on SGP.22 [2].

These Test Certificates are based on NIST P-256 and BrainpoolP256r1 curves and target the nominal test cases for the time being.

The certificates to be created, along with the relevant key pairs, are the following:

- One GSMA CI Certificate (CERT.CI.ECDSA) per curve
- One EUM Certificate (CERT.EUM.ECDSA) per curve
- Two SM-DP+ Certificates (CERT.DPauth.ECDSA and CERT.DPpb.ECDSA) per curve
- One SM-DP+ TLS Certificate (CERT.DP.TLS) per curve
- One eUICC Certificate (CERT.EUICC.ECDSA) per curve
- One SM-DS Certificate (CERT.DSauth.ECDSA) per curve
- One SM-DS TLS Certificate (CERT.DS.TLS) per curve

1.2 References

Ref	Document Number	Title
[1]	SGP.22	GSMA "RSP Technical specification" V2.1
[2]	SGP.23	GSMA "RSP Test Specification" v1.0
[3]	RFC5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

2 Tool chain for generation of the keys and certificates

This section describes the tools and the environment that have been used to generate the keys and the certificates described in this document.

2.1 OpenSSL

OpenSSL is an open source project that also provides a general-purpose cryptography library.

Information and documentation can be found here: https://www.openssl.org/.

Binaries can be downloaded here: https://wiki.openssl.org/index.php/Binaries.

The next section assumes that the tool has been installed and correctly configured in your environment.

The OpenSSL version used to generate the certificates in this document is 1.1.0e

2.2 Keys generation

The following command lines generate (randomly) a private key

- For NIST P-256 curve:

```
openssl ecparam -name prime256v1 -genkey -out <sk_file_name>
```

- For brainpoolP256r1 curve:

```
openssl ecparam -name brainpoolP256r1 -genkey -out <sk_file_name>
```

<sk_file_name> specifies the file name that will contain the generated private key (not encrypted) in the PEM form.

Note: The PEM form is the default format: it consists of the ASN.1 DER format base64 encoded with additional header and footer lines.

The complete description of the `Openssl ecparam` command can be found here: https://www.openssl.org/docs/man1.1.0/apps/ecparam.html

The following command line generates the related public key.

```
openssl ec -in <sk_file_name> -pubout -out <pk_file_name>
```

<sk_file_name> specifies the file name that contains the private key generated with the previous command line.

<pk_file_name> specifies the file name that will contain the generated public key in the PEM form.

The complete description of the `Openssl ec` command can be found here: https://www.openssl.org/docs/man1.1.0/apps/ec.html

2.3 CI Certificate Generation

The following command lines generate a root certificate like for the GSMA CI. The first command line generates the certificate in PEM format (Base64 encoded) and the second command line converts the same certificate from PEM format into DER (i.e. binary DER) encoded format.

```
openssl req -config <ca_configuration_file> -key <ca_sk_file_name> -new -x509 -days
<days> -sha256 -set_serial <serial> -extensions extend -out <cert_pem_file_name>

openssl x509 -in <cert_pem_file_name> -outform DER -out <cert_der_file_name>
```

<ca_configuration_file> is the configuration file that contains the attributes and extensions values of the CI certificate.

<ca_sk_file_name> specifies the file name that contains the CA private key in PEM format.

<serial> specifies the serial number to set in the certificate.

<days> specifies the number of days of validity to set in the certificate.

<cert_pem_file_name> specifies the file name that will contain the certificate in PEM format.

<cert_der_file_name> specifies the file name that will contain the certificate in DER format

The complete description of the `Openssl req` command can be found here:
https://www.openssl.org/docs/man1.1.0/apps/req.html

The complete description of the input data file format for <ca_configuration_file> specifying certificate extension can be found here:

https://www.openssl.org/docs/man1.1.0/apps/x509v3_config.html

2.4 Non-Root Certificate generation

The generation of a certificate starts with the generation of a Certificate Signing Request (CSR). The following command line generates this CSR.

```
openssl req -new -nodes -sha256 -config <input_csr_file_name> -key <sk_file_name> -
out <csr_file_name>
```

<input_csr_file_name> specifies the file name that contains the input data for CSR.

<sk_file_name> specifies the file name that contains the private key generated with the command described in section 2.2.

<csr_file_name> specifies the file name that will contain the generated CSR.

The complete description of the `Openssl req` command can be found here:
https://www.openssl.org/docs/man1.1.0/apps/req.html

The complete description of the input data file format for CSR can be found here:
https://www.openssl.org/docs/man1.1.0/apps/x509v3_config.html

The following command lines generate the certificate corresponding to a CSR. The first command line generates the certificate in PEM format (Base64 encoded) and the second command line converts the same certificate from PEM format into DER (i.e. binary DER) encoded format.

```
openssl x509 -req -in <csr_file_name> -CA <ca_cert_file_name> -CAkey
<ca_sk_file_name> -set_serial <serial> -days <days> -extfile <cert_ext_file_name> -
out <cert_pem_file_name>

openssl x509 -in <cert_pem_file_name> -outform DER -out <cert_der_file_name>
```

<csr_file_name> specifies the file name that contains the CSR generated with the previous command line.

<ca_cert_file_name> specifies the file name that contains the CA Certificate in PEM format.

<ca_sk_file_name> specifies the file name that contains the CA private key in PEM format related to the certificate indicated by <ca_cert_file_name>.

<serial> specifies the serial number to set in the certificate.

<days> specifies the number of days of validity to set in the certificate.

<cert_ext_file_name> specifies the file name that contains certificate extensions to set in the certificate.

<cert_pem_file_name> specifies the file name that will contain the certificate in PEM format.

<cert_der_file_name> specifies the file name that will contain the certificate in DER format

NOTE: As defined, the input CA certificate to generate the Non-Root Certificates SHALL be in PEM format, the following command will be used to convert from DER format to PEM format (whether the PEM format is not provided)

```
openssl x509 -inform der -in <cert_der_file_name> -out <cert_pem_file_name>
```

The complete description of the `Openssl x509` command can be found here: https://www.openssl.org/docs/man1.1.0/apps/x509.html

The complete description of the file format for specifying certificate extension can be found here: https://www.openssl.org/docs/man1.1.0/apps/x509v3_config.html

2.5 Certificate display

A certificate can be displayed with the following command lines.

```
openssl x509 -in <cert_pem_file_name> -text -noout
openssl x509 -in <cert_der_file_name> -inform der -text -noout
```

<cert_pem_file_name> specifies the file name that contains the certificate in PEM format.

<cert_der_file_name> specifies the file name that contains the certificate in DER format.

3 Test Certificates and keys

Please note that currently no CRLs are provided. It needs to be confirmed that the value contained in extension crlDistributionPoint will not lead to a problem with LPA/SM-DP+/SM-DS implementations.

3.1 Certificate Issuer

3.1.1 CI Certificate: definition of data to be signed

Field	Value
version	2
serialNumber	'00 B8 74 F3 AB FA 6C 44 D3'
signature	sha256ECDSA
Issuer	See 'subject'
Validity	12783 days (35 years)
Subject	cn = GSMA Test CI ou = TESTCERT o = RSPTEST c = IT
subjectPublicKeyInfo	algorithm.algorithm='1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters '1.2.840.10045.3.1.7' (prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey=[CI public key value]
Extension	(Sequence)
subjectKeyIdentifier extension	NIST: 'F5 41 72 BD F9 8A 95 D6 5C BE B8 8A 38 A1 C1 1D 80 0A 85 C3' Brainpool: 'C0 BC 70 BA 36 92 9D 43 B4 67 FF 57 57 05 30 E5 7A B8 FC D8'
keyUsage Extension	Certificate Signing, Off-line CRL Signing, CRL Signing (06)
certificatePolicies Extension	'2.23.146.1.2.1.0' (id-rspRole-ci)
basicConstraints Extension	CA = true
subjectAltName Extension	'2.999.1'
crlDistributionPoints Extension	[1]CRL Distribution Point Distribution Point Name: Full Name: URL=http://ci.test.gsma.com/CRL-A.crl [2]CRL Distribution Point Distribution Point Name: Full Name: URL=http://ci.test.gsma.com/CRL-B.crl

Table 1: CERT.CI.ECDSA

3.1.2 CI Keys and Certificate



3.1.3 Input data for generation

The SK.CI.ECDSA and PK.CI.ECDSA are generated using the command lines as described in section 2.2.

The CERT.CI.ECDSA is generated using the command lines described in section 2.3 with the following input data:

<ca_configuration_file>


CI-csr.cnf

<serial> set with value defined in section 3.1.1 for `serialNumber` data field.

<days> set with value defined in section 3.1.1 for `validity` data field.

3.2 eUICC

3.2.1 eUICC Certificate: definition of data to be signed

Field	Value
Version	2
serialNumber	'02 00 00 00 00 00 00 00 01'
signature	sha256ECDSA
Issuer	cn = EUM Test o = RSP Test EUM c = DE
Validity	2000000 days
Subject	cn = Test eUICC serialNumber = '89049032123451234512345678901235' (EID) o = RSP Test EUM c = DE
subjectPublicKeyInfo	algorithm.algorithm='1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters '1.2.840.10045.3.1.7' (prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey=[EUICC public key value] (see section 3.2.2)
Extension (Sequence)	
authorityKeyIdentifier Extension	<Value of CERT.EUM.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
subjectKeyIdentifier Extension	NIST: '36 F0 23 1A 9C 6D 76 93 C5 EF 4D FB 02 C2 09 D1 B4 2F 1D D0' Brainpool: C8 A6 4F 34 3B 85 B7 B0 57 8D C5 7F 8F 13 58 6D C8 04 ED 84

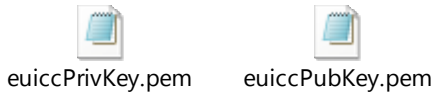
keyUsage Extension	Critical digitalSignature ('80')
certificatePolicies Extension	Critical '2.23.146.1.2.1.1' (id-rspRole-euicc)

Table 2: CERT.EUICC.ECDSA

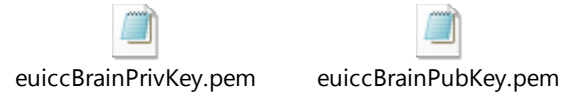
NOTE: OpenSSL tool does not allow the generation of Infinite duration certificates. For this reason, the eUICC certificate generated herein, only intended for test purposes, is not aligned with the SGP.14 specification. An eUICC certificate generated with another tool supporting this capability SHALL have the duration set to Infinite.

3.2.2 eUICC Keys and Certificate

NIST key pair:



Brainpool key pair:



CERT.EUICC.ECDSA:



3.2.3 Input data for generation

The SK.EUICC.ECDSA and PK.EUICC.ECDSA are generated using the command lines as described in section 2.2.

The CERT.EUICC.ECDSA is generated using the command lines described in section 2.4 with the following input data:

<input_csr_file_name>



<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.3.2 (file containing the CERT.EUM.ECDSA and SK.EUM.ECDSA respectively).

<serial> set with value defined in section 3.2.1 for `serialNumber` data field.

<days> set with value defined in section 3.2.1 for `validity` data field.


eUICC-ext.cnf

3.3 EUM

3.3.1 EUM Certificate: definition of data to be signed

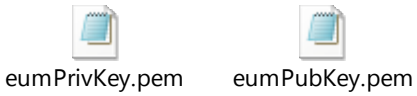
Field	Value
version	2
serialNumber	'12 34 56 78'
signature	algorithm = '1.2.840.10045.4.3.2' (sha256ECDSA)
Issuer	<Value of CERT.CI.ECDSA."subject" field>
validity	12410 days (34 years)
subject	cn = EUM Test o = RSP Test EUM c = DE
subjectPublicKeyInfo	algorithm.algorithm='1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey=[EUM public key value] (see section 3.3.2)
authorityKeyIdentifier Extension	<Value of CERT.CI.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
subjectKeyIdentifier Extension	NIST (prime256v1): DD:3D:A2:4D:35:0C:1C:C5:D0:AF:09:65:F4:0E:C3:4C:5E:E4:09:F1 Brainpool (brainpoolP256r1): 6F A1 E5 21 73 63 A8 22 BD ED 98 8A 1A 0D 0F F5 D7 62 0D B7
keyUsage Extension	Critical Certificate Sign (‘04’)
Certificate Policies	Critical '2.23.146.1.2.1.2' (id-rspRole-eum)
subjectAltName Extension	'2.999.5'
basicConstraints	Critical CA = true pathLenConstraint = 0

crlDistributionPoints Extension	[1]CRL Distribution Point Distribution Point Name: Full Name: URL=http://ci.test.gsma.com/CRL-B.crl
nameConstraints	Critical permittedSubtrees: id-at-organizationName: '2.5.4.10' organization name: "RSP Test EUM" UTF8String id-at-serialNumber: '2.5.4.5' iin: "89049032" PrintableString

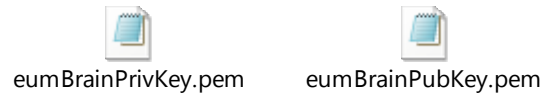
Table 3: CERT.EUM.ECDSA

3.3.2 EUM Keys and Certificate

NIST key pair:



Brainpool key pair:



CERT.EUM.ECDSA:



3.3.3 Input data for generation

The SK.EUM.ECDSA and PK.EUM.ECDSA are generated using the command lines as described in section 2.2.

The CERT.EUM.ECDSA is generated using the command lines described in section 2.4 with the following input data:

<input_csr_file_name>

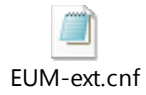


<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.1.2 (file containing the CERT.CI.ECDSA and SK.CI.ECDSA respectively).

<serial> set with value defined in section 3.3.2 for `serialNumber` data field.

<days> set with value defined in section 3.3.2 for `validity` data field.

<cert_ext_file_name>



3.4 SM-DP+

3.4.1 DPauth

3.4.1.1 SM-DP+ Certificate for Authentication: definition of data to be signed

Field	Value
Version	'2'
serialNumber	'100'
signature	algorithm = '1.2.840.10045.4.3.2' (sha256ECDSA)
Issuer	<Value of CERT.CI.ECDSA."subject" field>
Validity	1095 days (3 years)
Subject	o = 'ACME' cn = 'TEST SM-DP+'
subjectPublicKeyInfo	algorithm.algorithm='1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey= <PK.DPauth.ECDSA value> (see 3.4.1.2)
Extensions	(Sequence)
Extension for authorityKeyIdentifier	<Value of CERT.CI.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
Extension for subjectKeyIdentifier	NIST: 'BD 5A 82 CC 1A 96 60 21 18 BA 75 60 A1 FF 83 A7 8B 21 0B E5' Brainpool: '79 A4 BD 4D 78 FF 47 34 BC 60 45 CF 91 96 24 4A 1F B8 4B EB'
Extension for keyUsage	Digital Signature ('80')
Extension for certificatePolicies	'2.23.146.1.2.1.4' (id-rspRole-dp-auth)

Extension for subjectAltName	'2.999.10'
Extension for crlDistributionPoints	<Value of CERT.CI.ECDSA."crlDistributionPoints" field>

Table 4: CERT.DPauth.ECDSA

3.4.1.2 SM-DP+ Keys and Certificate

NIST key pair:


DPauth.key.pub_NIST.pem


DPauth.key.priv_NIST.pem

Brainpool key pair:


DPauth.Brain256.key.pub.pem


DPauth.Brain256.key.priv.pem

CERT.DPauth.ECDSA:


DPauth_NIST.der


DPauth.Brain256.der

3.4.1.3 Input data for generation

The SK.DPauth.ECDSA and PK.DPauth.ECDSA are generated using the command lines as described in section 2.2.

The CERT.DPauth.ECDSA is generated using the command lines described in section 2.4 with the following input data:

<input_csr_file_name>


DP.csr.cnf

<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.1.2 (file containing the CERT.CI.ECDSA and SK.CI.ECDSA respectively).

<serial> set with value defined in section 3.4.1.1 for `serialNumber` data field.

<days> set with value defined in section 3.4.1.1 for `validity` data field.

<cert_ext_file_name>


DPauth-ext.cnf

3.4.2 DPpb

3.4.2.1 SM-DP+ Certificate for Profile Binding: definition of data to be signed

Field	Value
Version	'2'
serialNumber	'101'
Signature	algorithm = '1.2.840.10045.4.3.2' (sha256ECDSA)
Issuer	<Value of CERT.CI.ECDSA."subject" field>
Validity	1095 days (3 years)
Subject	o = 'ACME' cn = 'TEST SM-DP+'
subjectPublicKeyInfo	algorithm.algorithm='1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey= <PK.DPpb.ECDSA value> (see 3.4.2.2)
Extensions	(Sequence)
Extension for authorityKeyIdentifier	< Value of CERT.CI.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
Extension for subjectKeyIdentifier	NIST (prime256v1): 'E6 EA F7 1E E0 FB 94 30 EC CD 1E BB 42 1F 88 14 37 C1 32 63' Brainpool (brainpoolP256r1): 'A8 C6 8D F4 49 EB 71 EC 72 3E AC 13 2E 40 E4 B6 F5 46 44 FE'
Extension for keyUsage	Digital Signature ('80')
Extension for certificatePolicies	'2.23.146.1.2.1.5' (id-rspRole-dp-pb)
Extension for subjectAltName	'2.999.10'
Extension for crlDistributionPoints	<Value of CERT.CI.ECDSA."crlDistributionPoints" field>

Table 5: CERT.DPpb.ECDSA

3.4.2.2 SM-DP+ Keys and Certificate

NIST key pair:


DPpb.key.pub_NIST.pem


DPpb.key.priv_NIST.pem

Brainpool key pair:


DPpb.Brain256.key.pub.pem


DPpb.Brain256.key.priv.pem

CERT.DPpb.ECDSA:


DPpb_NIST.der


DPpb.Brain256.der

3.4.2.3 Input data for generation

The SK.DPpb.ECDSA and PK.DPpb.ECDSA are generated using the command lines as described in section 2.2.

The CERT.DPpb.ECDSA is generated using the command lines described in section 2.4 with the following input data:

<input_csr_file_name>


DP.csr.cnf

<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.1.2 (file containing the CERT.CI.ECDSA and SK.CI.ECDSA respectively).

<serial> set with value defined in section 3.4.2.1 for serialNumber data field.

<days> set with value defined in section 3.4.2.1 for validity data field.

<cert_ext_file_name>


DPpb-ext.cnf

3.4.3 TLS

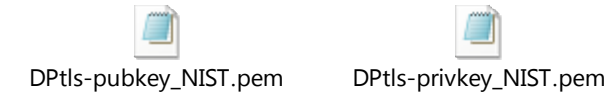
3.4.3.1 SM-DP+ TLS Certificate: definition of data to be signed

Field	Value
version	2
serialNumber	'9'
signature	algorithm = '1.2.840.10045.4.3.2' (sha256ECDSA)
issuer	<Value of CERT.CI.ECDSA."subject" field>
validity	1095 days (3years)
subject	o = 'ACME' cn = 'smdp-plus.test.gsma.com'
subjectPublickeyInfo	algorithm.algorithm= '1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (Prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey = < PK.DP.TLS value> (see 3.4.6)
Extensions	(Sequence)
Extension for authorityKeyIdentifier	<Value of CERT.CI.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
Extension for subjectKeyIdentifier	NIST (prime256v1): '27 FE F1 F2 29 18 7E C7 83 ED F6 E0 29 64 A4 51 8D 57 D4 A9' Brainpool (brainpoolP256r1): '3D 33 09 83 F3 9F CC 5B D2 E4 AD 68 A6 19 A7 47 48 AE 8B 9D'
Extension for keyUsage	Critical digitalSignature ('80')
Extension for certificatePolicies	'2.23.146.1.2.1.3' (id-rspRole-dp-tls)
Extension for extendedKeyUsage	Critical TLS Web Server Authentication TLS Client Authentication
Extension for subjectAltName	DNS= smdp-plus.test.gsma.com SM-DP+OID = '2.999.10'
Extension for crlDistributionPoints	<Value of CERT.CI.ECDSA."crlDistributionPoints" field>

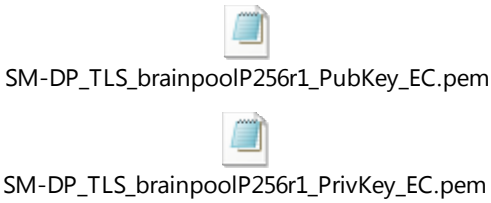
Table 6: CERT.DP.TLS

3.4.3.2 SM-DP+ TLS Keys and Certificate

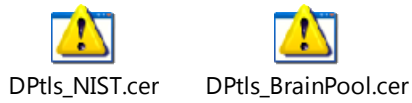
NIST key pair:



Brainpool key pair:



CERT.DP.TLS:



3.4.3.3 Input data for generation

Command lines for the generation of the SK.DP.TLS and the corresponding PK.DP.TLS for NIST P-256 curve.

```
Openssl ecparam –name prime256v1 –genkey –out DPtls-privkey.pem
openssl ec –in DPtls-privkey.pem –pubout –out DPtls-pubKey.pem
```

Command lines for the generation of the CERT.DP.TLS.

The first command line generates a Certificate Signing Request (CSR) that is used as an input data for the second and third command line. The second command line generates the CERT.DP.TLS in (Base64) encoded PEM format and the third command line converts the DER format into CER (i.e. binary DER) encoded format.

```
Openssl req –new –nodes –sha256 –config DPtls.cnf –key DPtls-privkey.pem –out
DPtls.csr

openssl x509 –req –in DPtls.csr –CA .\Test_CI.cer –Cakey .\TestCI.pem –set_serial
0x09 –days 1095 –sha256 –extfile DPtls_ext.cnf –out DPtls.pem

openssl x509 –in DPtls.pem –outform DER –out DPtls.cer
```

The following configuration files are used for the generation of the certificate request and the certificate (in the second and third command line):


DPtls_csr.cnf


DPtls_ext.cnf

3.5 SM-DS

3.5.1 DSauth

3.5.1.1 SM-DS Certificate for Authentication: definition of data to be signed

Field	Value
version	2
serialNumber	'7495'
signature	algorithm = '1.2.840.10045.4.3.2' (sha256ECDSA)
issuer	<Value of CERT.CI.ECDSA."subject" field>
validity	1095 days (3 years)
subject	o = 'ACME' cn = 'TEST SM-DS'
subjectPublicKeyInfo	algorithm.algorithm= '1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (Prime256v1) or '1.3.36.3.3.2.8.1.1.7' (brainpoolP256r1) subjectPublicKey = < PK.Dsauth.ECDSA value> (see 3.5.1.2)
Extensions	(Sequence)
Extension for Authority Key Identifier	<Value of CERT.CI.ECDSA."subjectKeyIdentifier" field> for prime256v1 or brainpoolP256r1
Extension for subjectKeyIdentifier	NIST (prime256v1): 'C1 F4 06 4B 3B 25 8A FB 61 38 8B 3F F2 EE 6A 61 E2 C4 4D 72' Brainpool (brainpoolP256r1): 'F0 5F 0B 54 AE E8 AE 01 08 F0 1D EF 54 8E D9 85 97 14 DD 48'
KeyUsage Extension	Digital Signature ('80')
Extension for Certificate Policy	'2.23.146.1.2.1.7' (id-rspRole-ds-auth)
Extension for subjectAltName	SM-DS OID = '2.999.15'
Extension for CRL Distribution Points	<Value of CERT.CI.ECDSA."crlDistributionPoints" field>

Table 7: CERT.Dsauth.ECDSA

3.5.1.2 SM-DS Keys and Certificate

NIST key pair:


SM-DS_prime256v1_PubKey_EC.pem


SM-DS_prime256v1_PrivKey_EC.pem

Brainpool key pair:


SM-DS_brainpoolP256r1_PrivKey_EC.pem


SM-DS_brainpoolP256r1_PubKey_EC.pem

CERT.Dsauth.ECDSA:


DSauth_NIST.der


DSAuth_BrainPool.cer

3.5.1.3 Input data for generation

The SK.Dsauth.ECDSA and PS.Dpauth.ECDSA are generated using the command lines as described in section 2.2.

The CERT.Dsauth.ECDSA is generated using the command lines described in section 2.4 with the following input data:

< DSAuth.csr >


DSauth-csr.ext

<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.1.2 (file containing the CERT.CI.ECDSA and SK.CI.ECDSA respectively).

<serial> set with value defined in section 3.5.1.1 for serialNumber data field.

<days> set with value defined in section 3.5.1.1 for validity data field.

<DSAuth.ext>


DSauth-ext.cnf

3.5.2 TLS

3.5.2.1 SM-DS TLS Certificate: definition of data to be signed

Field	Value
Version	2
serialNumber	'1223334444'
Signature	SHA256ECDSA
Issuer	<Value of CERT.CI.ECDSA.”subject” field>
Validity	1095 days (3years)
Subject	o = ‘RSPTEST’ cn = ‘smds.test.gsma.com’
subjectPublicKeyInfo	algorithm.algorithm= '1.2.840.10045.2.1' (id-ecPublicKey) algorithm.parameters= '1.2.840.10045.3.1.7' (Prime256v1) or '1.3.36.3.3.2.8.1.1.7' (BrainpoolP256r1) subjectPublicKey = < PK.DS.TLS value>
Extensions	(Sequence)
Extension for Authority Key Identifier	<Value of CERT.CI.ECDSA.”subjectKeyIdentifier” field> for Prime256v1 or BrainpoolP256r1
Extension for Subject Key Identifier	NIST: 'A0 36 C1 62 75 35 1E C7 B0 15 53 A1 3F 83 E2 8D 44 00 BD 0A' Brainpool: ‘73 99 CA C7 B1 5F AB 2F F9 33 CF 2D 22 15 E4 84 4A DE F8 05’
Extension for Key usage	Critical digitalSignature (‘80’)
Extension for Certificate Policies	‘2.23.146.1.2.1.6’ (id-rspRole-ds-tls)
Extension for Extended Key usage	Critical TLS Web Server Authentication , TLS Web Client Authentication
Extension for subjectAltName	DNS= smds.test.gsma.com SM-DS OID = ‘2.999.15’
Extension for CRL Distribution Points	<Value of CERT.CI.ECDSA.”crlDistributionPoints” field>

Table 8: CERT.DS.TLS

3.5.2.2 SM-DS TLS Keys and Certificate

NIST key pair:


DStls-privkey_NIST.pem


DStls-pubkey_NIST.pem

Brainpool key pair:


TestDSTLS_PKbrainpool.pem


TestDSTLS_SKbrainpool.pem

CERT.DS.TLS:


DSTLS-cert_NIST.der


DSTLS-cert-brainpool.der

3.5.2.3 Input data for generation

The SK.DS.TLS and PK.DS.TLS are generated using the command lines as described in section 2.2.

The CERT.DS.TLS is generated using the command lines described in section 2.4 with the following input data:

<input_csr_file_name>


DStls_csr.cnf

<ca_cert_file_name> and <ca_sk_file_name>: files generated in section 3.1.2 (file containing the CERT.CI.ECDSA and SK.CI.ECDSA respectively).

<serial> set with value defined in section 3.5.2.1 for `serialNumber` data field.

<days> set with value defined in section 3.5.2.1 for `validity` data field.

<cert_ext_file_name>


DStls_ext.cnf

Annex A Document Management (Informative)

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
v1.0	9 June 2017	New PRD Publication	PSMC	Yolanda Sanz GSMA

Other Information

Type	Description
Document Owner	Yolanda Sanz / GSMA
Editor / Company	Guido Abate / STMicroelectronics

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.