CERTIFICATS : CERTIFICATE THUMBPRINT

Ne pas le confondre avec Public Key Thumbprint (dnQualifier)

PRÉFACE

Dans les KDM dans les CipherValue, on va vous demander un **Certificate Thumbprint**.

Le Certificate Thumbprint est l'empreinte SHA-1 de la partie To-Be-Signed (tbsCertificate) du certificat en excluant l'entête DER.

Normalement, le **Certificate Thumbprint** ne sera que représenté que sous sa forme **SHA1** en binaire (utilisé dans le **CipherValue**), mais si une représentation lisible doit être effectuée, il doit être en Base64.

```
CertificateThumbprint            =          SHA1 ( tbsCertificate # withoutDERHeader )
CertificateThumbprint (printable) = Base64 ( SHA1 ( tbsCertificate # withoutDERHeader ) )
```

Pour comprendre ce qu'est le tbsCertificate, voyons la représentation normée d'un certificat complet :

```
Certificate  ::=  SEQUENCE  {
    tbsCertificate          TBSCertificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signatureValue          BIT STRING
}
```

Si nous analysons cette normalisation, un certificat est composé de :

- Une structure de données nommée **TBSCertificate**
- Suivi d'un identifiant pour l'algorithme utilisé pour la signature (**signatureAlgorithm**)
- Suivi de la signature en elle-même (**signatureValue**)

La partie qui nous intéresse est la partie verte : **TBSCertificate**.

Analysons maintenant la structure de **TBSCertificate** qui est normée de la sorte :

```
TBSCertificate ::= SEQUENCE {
    version                 Version DEFAULT v1(0),
    serialNumber            CertificateSerialNumber,
    signature               AlgorithmIdentifier,
    issuer                  Name,
    validity                Validity,
    subject                 Name,
    subjectPublicKeyInfo    SubjectPublicKeyInfo,
    issuerUniqueID          IMPLICIT UniqueIdentifier OPTIONAL,
    subjectUniqueID         IMPLICIT UniqueIdentifier OPTIONAL,
    extensions              Extensions OPTIONAL
}
```

Vous remarquez des éléments que vous reconnaissez et que nous avons vu en tout début de ce chapitre.

Si nous revoyons notre sortie classique et nous mettons en avant notre TBSCertificate, vous verrez rapidement ce que compose ce dernier :

```
$ openssl x509 -in certificate.pem -text
Data:
    Version: 3 (0x2)
    Serial Number: 643 (0x283)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: O = DC2.SMPTE.DOREMILABS.COM, OU = DC.DOREMILABS.COM, CN = .DC.DMS.DC2.SMPTE, dnQualifier = "+LLvuYNO4Y
    Validity
        Not Before: Jan  1 00:00:00 2007 GMT
        Not After : Dec 31 23:59:59 2025 GMT
    Subject: O = DC2.SMPTE.DOREMILABS.COM, OU = DC.DOREMILABS.COM, CN = CS.DMSJP2K-80119.DC.DC2.SMPTE, dnQualifier
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:bc:62:a5:fb:33:4f:73:58:2a:6a:03:37:2b:52:
                62:4e:17:1a:41:6f:6e:f4:c3:98:b5:32:4c:4d:54:
                62:c4:e7:ee:e6:c1:88:39:80:05:7f:a1:66:49:fe:
                55:9d:e1:06:1b:db:5d:fe:23:7a:1a:99:48:d5:ee:
                6e:a5:3b:e4:cb:0d:42:4e:68:ae:02:9d:e9:a9:ca:
                8b:84:cf:2f:c8:f9:ed:ce:5a:08:09:33:71:7b:d0:
                08:8a:e3:a7:25:03:b5:92:12:c6:51:0d:69:80:3e:
                4c:be:97:f2:6b:fe:08:8a:a9:ea:f2:ea:51:f3:83:
                e8:2e:79:ec:10:ab:e8:c5:eb:97:6b:58:8b:ac:2c:
                83:67:29:0f:ed:86:e7:f1:4e:66:bb:37:40:aa:bf:
                58:46:e1:73:17:36:db:ab:c1:5c:af:3d:6e:3e:1c:
                8a:78:ad:22:eb:e6:50:55:d7:d3:f3:1c:b8:06:e3:
                46:41:cb:9c:8a:63:c6:a0:89:ae:fb:6c:9f:35:b1:
                1c:7c:54:1f:7c:30:39:6c:6c:d1:db:f5:c3:25:32:
                c4:6a:a3:70:7b:f5:97:67:21:a8:91:9b:48:47:39:
                85:25:81:9b:e1:9c:be:a5:81:96:20:ae:6e:9a:e8:
                63:34:51:13:b9:f3:48:e3:c9:b2:6c:d6:6d:7a:5a:
                63:23
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Key Usage:
            Digital Signature, Key Encipherment, Data Encipherment
        X509v3 Subject Key Identifier:
            49:01:58:49:2A:96:C2:37:9F:A6:9A:9A:B0:C2:66:7D:D9:92:EA:52
        X509v3 Authority Key Identifier:
            keyid:F8:B2:EF:B9:83:4E:E1:80:49:4A:9F:49:8E:69:6F:F2:88:A9:A7:34
            DirName:/O=DC2.SMPTE.DOREMILABS.COM/OU=DC.DOREMILABS.COM/CN=.DMS.DC2.SMPTE/dnQualifier=RQ\/53RmuLsbzgfF
            serial:02
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
    3c:1a:59:e9:39:20:f5:36:60:d8:b6:a6:2d:a3:82:3c:4e:a2:
    9e:00:6f:aa:6b:10:1d:1e:55:e1:bf:7d:b0:d5:3f:12:f7:46:
    2f:67:89:8d:91:70:e1:82:ec:c5:a1:26:3e:9a:48:18:57:4c:
    20:4f:90:24:7a:c4:0f:96:0b:68:9f:62:d5:5b:d3:6c:3d:63:
    35:fa:dc:95:6e:12:6d:ce:be:9a:54:0a:14:2e:af:38:3e:7f:
    82:8d:e1:2d:80:bc:03:9c:3d:d9:e6:bb:e6:25:fb:ed:2b:83:
    d0:30:83:d4:62:2c:e8:52:f7:10:64:ab:31:70:b9:f4:71:4a:
    e1:a8:67:22:4c:5c:53:28:55:ef:90:a7:d7:8b:a3:68:e4:29:
    88:ef:b3:07:1a:ff:55:a8:bf:3c:36:68:cb:a2:92:9b:4c:98:
    24:48:7d:ee:ae:3e:bd:06:10:95:15:92:3f:57:05:f4:88:cb:
    ba:ca:d8:05:38:d4:df:47:fb:af:28:0e:47:8b:dc:a8:bf:31:
    9c:d4:21:62:9a:c1:94:90:67:f8:a4:b7:16:a3:4a:b6:b5:28:
    1d:31:74:62:d2:e5:e0:8e:65:f7:4a:1c:b6:5f:af:b5:03:46:
    5b:1a:b8:c5:4d:f7:0e:ef:6a:5c:e0:57:04:4f:61:79:27:c6:
    dc:2b:26:57
```

Nous écartons donc l'entête ASN.1 DER (4 octets)

Si on utilise `openssl asn1parse`, nous verrons plus en détail chaque partie de notre certificat :

```
$ openssl asn1parse -in certificate.pem
    0:d=0  hl=4 l=1146 cons: SEQUENCE
    4:d=1  hl=4 l= 866 cons: SEQUENCE
    8:d=2  hl=2 l=   3 cons: cont [ 0 ]
   10:d=3  hl=2 l=   1 prim: INTEGER           :02
   13:d=2  hl=2 l=   2 prim: INTEGER           :0283
   17:d=2  hl=2 l=  13 cons: SEQUENCE
   19:d=3  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
   30:d=3  hl=2 l=   0 prim: NULL
   32:d=2  hl=3 l= 130 cons: SEQUENCE
   35:d=3  hl=2 l=  33 cons: SET
   37:d=4  hl=2 l=  31 cons: SEQUENCE
   39:d=5  hl=2 l=   3 prim: OBJECT            :organizationName
   44:d=5  hl=2 l=  24 prim: PRINTABLESTRING   :DC2.SMPTE.DOREMILABS.COM
   70:d=3  hl=2 l=  26 cons: SET
   72:d=4  hl=2 l=  24 cons: SEQUENCE
   74:d=5  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
   79:d=5  hl=2 l=  17 prim: PRINTABLESTRING   :DC.DOREMILABS.COM
   98:d=3  hl=2 l=  26 cons: SET
  100:d=4  hl=2 l=  24 cons: SEQUENCE
  102:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  107:d=5  hl=2 l=  17 prim: PRINTABLESTRING   :.DC.DMS.DC2.SMPTE
  126:d=3  hl=2 l=  37 cons: SET
  128:d=4  hl=2 l=  35 cons: SEQUENCE
  130:d=5  hl=2 l=   3 prim: OBJECT            :dnQualifier
  135:d=5  hl=2 l=  28 prim: PRINTABLESTRING   :+LLvuYNO4YBJSp9Jjmlv8oippzQ=
  165:d=2  hl=2 l=  30 cons: SEQUENCE
  167:d=3  hl=2 l=  13 prim: UTCTIME           :070101000000Z
  182:d=3  hl=2 l=  13 prim: UTCTIME           :251231235959Z
  197:d=2  hl=3 l= 142 cons: SEQUENCE
  200:d=3  hl=2 l=  33 cons: SET
  202:d=4  hl=2 l=  31 cons: SEQUENCE
  204:d=5  hl=2 l=   3 prim: OBJECT            :organizationName
  209:d=5  hl=2 l=  24 prim: PRINTABLESTRING   :DC2.SMPTE.DOREMILABS.COM
  235:d=3  hl=2 l=  26 cons: SET
  237:d=4  hl=2 l=  24 cons: SEQUENCE
  239:d=5  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
  244:d=5  hl=2 l=  17 prim: PRINTABLESTRING   :DC.DOREMILABS.COM
  263:d=3  hl=2 l=  38 cons: SET
  265:d=4  hl=2 l=  36 cons: SEQUENCE
  267:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  272:d=5  hl=2 l=  29 prim: PRINTABLESTRING   :CS.DMSJP2K-80119.DC.DC2.SMPTE
  303:d=3  hl=2 l=  37 cons: SET
  305:d=4  hl=2 l=  35 cons: SEQUENCE
  307:d=5  hl=2 l=   3 prim: OBJECT            :dnQualifier
  312:d=5  hl=2 l=  28 prim: PRINTABLESTRING   :SQFYSSqWwjefppqasMJmfdmS6lI=
  342:d=2  hl=4 l= 290 cons: SEQUENCE
  346:d=3  hl=2 l=  13 cons: SEQUENCE
  348:d=4  hl=2 l=   9 prim: OBJECT            :rsaEncryption
  359:d=4  hl=2 l=   0 prim: NULL
  361:d=3  hl=4 l= 271 prim: BIT STRING
  636:d=2  hl=3 l= 235 cons: cont [ 3 ]
  639:d=3  hl=3 l= 232 cons: SEQUENCE
  642:d=4  hl=2 l=  12 cons: SEQUENCE
  644:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
  649:d=5  hl=2 l=   1 prim: BOOLEAN           :255
  652:d=5  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
  656:d=4  hl=2 l=  11 cons: SEQUENCE
  658:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
  663:d=5  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:030204B0
  669:d=4  hl=2 l=  29 cons: SEQUENCE
  671:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Identifier
  676:d=5  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:0414490158492A96C2379FA69A9AB0C2667DD992EA52
  700:d=4  hl=3 l= 171 cons: SEQUENCE
  703:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Identifier
  708:d=5  hl=3 l= 163 prim: OCTET STRING      [HEX DUMP]:3081A08014F8B2EFB9834EE180494A9F498E696FF288A9A734A18184A
  874:d=1  hl=2 l=  13 cons: SEQUENCE
  876:d=2  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
  887:d=2  hl=2 l=   0 prim: NULL
  889:d=1  hl=4 l= 257 prim: BIT STRING
```

La partie en vert est la partie **To-Be-Signed** (tbsCertificate). C'est cette partie qu'on va récupérer pour générer

notre **Certificate Thumbprint**.

Vous remarquez les zones en rouge, ce sont nos parties que nous ecarterons lors du calcul de l'empreinte. La première partie en rouge (SEQUENCE) est notre entête ASN.1 DER de notre certificat que nous écartons. Et la seconde partie en rouge (à partir de l'offset 874 : **SEQUENCE**, **OBJECT** *sha256WithRSAEncryption*, **NULL**, **BIT STRING**) sont l'identifiant de l'algorithme (OID) utilisé pour la signature et notre fameuse signature.

> **Attention**, cet exemple avec l'offset 874 ne marche qu'avec ce certificat, le length depend du certificat !

Pour écarter l'entête ASN.1 DER, nous devons donc commencer à offset 4 et nous arrêter à l'offset 874 pour éviter toute la partie Signature :

```
openssl asn1parse -in certificate.pem \
       -offset 4 \
       -length 870 \      # offset 874 - offset 4
       -noout -out - \
       | openssl sha1
SHA1(stdin)= 09d53df013c07fa4341fded0eb57cf7a807a687d
```

C'est cette valeur que nous utiliserons pour le **Certificate Thumbprint** dans **CipherValue** d'un KDM.

Maintenant, effectuons la même mais en Python :)

```
#!/usr/bin/env python3
import hashlib
import base64

# Le certificat public entier (format PEM ASN.1)
pubcert = """MIIEejCCA2KgAwIBAgICAoMwDQYJKoZIhvcNAQELBQAwgYIxITAfBgNVBAoTGERD
Mi5TTVBURS5ET1JFTUlMQUJTLkNPTTEaMBgGA1UECxMRREMuRE9SRU1JTEFCUy5D
T00xGjAYBgNVBAMTES5EQy5ETVMuREMyLlNNUFRFMSUwIwYDVQQuExwrTEx2dVlO
TzRZQkpTcDlKam1sdjhvaXBwelE9MB4XDTA3MDEwMTAwMDAwMFoXDTI1MTIzMTIz
NTk1OVowgY4xITAfBgNVBAoTGERDMi5TTVBURS5ET1JFTUlMQUJTLkNPTTEaMBgG
A1UECxMRREMuRE9SRU1JTEFCUy5DT00xJjAkBgNVBAMTHUNTLkRNU0pQMkstODAx
MTkuREMuREMyLlNNUFRFMSUwIwYDVQQuExxTUUZZU1NxV3dqZWZwcHFhc01KbWZk
bVM2bEk9MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvGKl+zNPc1gq
agM3K1JiThcaQW9u9MOYtTJMTVRixOfu5sGIOYAFf6FmSf5VneEGG9td/iN6GplI
1e5upTvkyw1CTmiuAp3pqcqLhM8vyPntzloICTNxe9AIiuOnJQO1khLGUQ1pgD5M
vpfya/4Iiqnq8upR84PoLnnsEKvoxeuXa1iLrCyDZykP7Ybn8U5muzdAqr9YRuFz
Fzbbq8Fcrz1uPhyKeK0i6+ZQVdfT8xy4BuNGQcucimPGoImu+2yfNbEcfFQffDA5
bGzR2/XDJTLEaqNwe/WXZyGokZtIRzmFJYGb4Zy+pYGWIK5umuhjNFETufNI48my
bNZtelpjIwIDAQABo4HrMIHoMAwGA1UdEwEB/wQCMAAwCwYDVR0PBAQDAgSwMB0G
A1UdDgQWBBRJAVhJKpbCN5+mmpqwwmZ92ZLqUjCBqwYDVR0jBIGjMIGggBT4su+5
g07hgElKn0mOaW/yiKmnNKGBhKSBgTB/MSEwHwYDVQQKExhEQzIuU01QVEUuRE9S
RU1JTEFCUy5DT00xGjAYBgNVBAsTEURDLkRPUkVNSUxBQlMuQ09NMRcwFQYDVQQD
Ew4uRE1TLkRDMi5TTVBURTElMCMGA1UELhMcUlEvNTNSbXVMc2J6Z2ZQWEdsUllt
SnJ1d01zPYIBAjANBgkqhkiG9w0BAQsFAAOCAQEAPBpZ6Tkg9TZg2LamLaOCPE6i
ngBvqmsQHR5V4b99sNU/EvdGL2eJjZFw4YLsxaEmPppIGFdMIE+QJHrED5YLaJ9i
1VvTbD1jNfrclW4Sbc6+mlQKFC6vOD5/go3hLYC8A5w92ea75iX77SuD0DCD1GIs
6FL3EGSrMXC59HFK4ahnIkxcUyhV75Cn14ujaOQpiO+zBxr/Vai/PDZoy6KSm0yY
JEh97q4+vQYQlRWSP1cF9IjLusrYBTjU30f7rygOR4vcqL8xnNQhYprBlJBn+KS3
FqNKtrUoHTF0YtLl4I5l90octl+vtQNGWxq4xU33Du9qXOBXBE9heSfG3CsmVw=="""

# Unwrapping PEM ASN.1 (ascii) to DER ASN.1 (binary)
der = base64.b64decode(pubcert)

# tbsCertificate (remove header and footer)
# -- for this certificate only
der = der[4:874]

# Create SHA1 fingerprint from pubkey DER ASN.1 format
engine = hashlib.sha1()
engine.update(der)
sha1 = engine.digest()

# Wrap on base64
hash = base64.b64encode(sha1)

# That's it ! :)
print("CertificateThumbprint :")
print("SHA1   = %s", sha1.hex())
print("BASE64 = %s" % , hash.decode('utf8'))
```

Dont voici la sortie avec notre certificat public de notre encodeur :

```
$ ./certificateThumbprint.py
CertificateThumbprint :
SHA1   = 09d53df013c07fa4341fded0eb57cf7a807a687d
BASE64 = CdU98BPAf6Q0H97Q61fPeoB6aH0=
```

La valeur binaire SHA1 sera utilisée pour la valeur du champ **Certificat Thumbprint** de notre **CipherValue** dans notre KDM.

CONCLUSION

Tout comme le Public Key Thumbprint (dnQualifier), le **Certificate Thumbprint** est un élément important dans notre workflow cryptographique.

CHAPITRES ANNEXES

- Certificats - Les bases
- Certificats - Les champs (fields) d'un certificat x509 DCI
- Certificats - Identity Attributes - les attributs et leurs rôles
- Certificats - La chaîne de certification
- Certificats - Certificate Thumbprint - l'empreinte du certificat ⬅ vous êtes ici
- Certificats - Public Key Thumbprint (dnQualifier) - l'empreinte de la clef public
- Certificats - Création : Nos propres certificats
- RSA - La cryptographie asymétrique pour le chiffrement et les signatures.

RÉFÉRENCES

- https://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt
- https://www.rfc-editor.org/rfc/rfc3280#section-4.1