Certificate / Signature

Additional information and examples on header and signature fields

Argenta Bank- en Verzekeringsgroep nv, Belgiëlei 49-53, 2018 Antwerpen

PSD2 Compliance Berlin Group - Certificates and signatures

Sources

The information in this document is meant as a extension to go into more details on the signatures aspects decribed in the following general guides:

Argenta implementation guide

NextGenPSD2 XS2A Framework: Implementation guidelines

Certificates

Accepted policies

For a certificate to be valid it must contain at least 1 of the following accepted policies:

- QWAC:
  - 0.4.0.194112.1.4
  - 0.4.0.19495.3.1
- QSEAL:
  - 0.4.0.194112.1.1

> **Note:** On our Sandbox environment, this policy check is not activated

Signing

With our PSD2 APIs the TPP needs to sign the request and send the tpp-certificate in header, which will ensure the data integrity of the request.

Required headers

As part of signature model, TPP needs to send 3 specific headers:

- **Digest**: It contains a hash of the message body. SHA-256 and SHA-512 are the only supported hash algorithms that can be used to calculate the Digest. Digest can be computed by taking the byte array of

json body and then creating a message digest of it using SHA-256 or SHA-512 algorithm, then convert in Base64 encoding value.

> **Note:** For the GET and DELETE endpoints, the digest will be equal to a byte array of 0 size.

- **Tpp-Signature-Certificate**: The certificate used for signing the request, in PEM encoding. The certificate can be put with or without the "BEGIN" and "END".

> **Note:** This string should not contain any new lines. You might need to remove End-Of-Line characters from your PEM file before adding the text to the header.

- **Signature**: A signature of the request by the TPP on application level. Signature can be computed by concatenating key=value of following parameters: keyId, algorithm, headers (digest and x-request-id) and signature.

Signature header

Required parameters

Following are the mandatory parameters that the signature header needs to contain:

- **keyId:** Used to look up the components that the server needs to validate the signature and certificate against. It is formatted as follows: keyId="SN=XXX,CA=YYYYYYYYYYYYYYY", where:
    - "XXX" is the serial number of the certificate in hexadecimal coding given in the TPP-Signature-CertificateHeader
    - "YYYYYYYYYYYYYYY" is the full Distinguished Name of the Certification Authority having produced this certificate.
- **algorithm:** Used to specify the digital signature algorithm to use when generating the signature. The algorithm must identify the same algorithm for the signature as presented in the certificate (Element "TPPSignature-Certificate") of this Request. It must identify SHA-256 or SHA-512 as Hash algorithm.

> **Note:** Most common values are: "rsa-sha256" & "rsa-sha512"

- **headers:** Used to specify the list of HTTP headers included when generating the signature for the message. Note that the list order is important, and MUST be specified in the order the HTTP header field-value pairs are concatenated together during signing. Must include "digest" header and "x-request-id" header, no other entries may be included.

> **Note:** Value must this always be: "digest x-request-id"

- **signature:** This is a base 64 encoded digital signature, as described in RFC 4648 [RFC4648], Section 4. The client uses the `algorithm` and `headers` signature parameters to form a canonicalised `signing string`. This `signing string` is then signed with the key associated with `keyId` and the algorithm corresponding to `algorithm`. The `signature` parameter is then set to the base 64 encoding of the signature

> **Note:** Private key used for signing the request shouldn't contain any new lines (End-Of-Line characters) and it should be used as one single string.

Calculation algorithm

Following is a code example (using Apache Groovy jmeter) of a how a working signature calculation algorithm can be designed:

```
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.Signature;
import java.util.Base64;
import java.time.LocalDateTime;
import java.security.spec.MGF1ParameterSpec;
import java.security.spec.PSSParameterSpec;
import java.security.MessageDigest;
import org.apache.jmeter.config.Argument;
import org.apache.jmeter.config.Arguments;

String bodyJson = "";
Arguments arguments = sampler.getArguments();
for (int i=0; i<arguments.getArgumentCount(); i++)
{
    Argument argument = arguments.getArgument(i);
    bodyJson = argument.getValue();
}

MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
messageDigest.update(bodyJson.getBytes());
String digest = "SHA-256=" + Base64.getEncoder().encodeToString(messageDigest.digest());

String headers = "digest x-request-id";
PrivateKey privateKey = KeyFactory.getInstance("RSA")
                                  .generatePrivate(new PKCS8EncodedKeySpec(
                                     Base64.getDecoder().decode("${__P(tppShaPrivateKey)}")));

Signature sha256withRSA= Signature.getInstance("SHA256withRSA");
sha256withRSA.initSign(privateKey);

String unencryptedSignature = "digest: " + digest + "\nx-request-id: ${request-id-consent}";
sha256withRSA.update(unencryptedSignature.getBytes("UTF-8"));

String encryptedSignature = new String(Base64.getEncoder().encode(sha256withRSA.sign()));
String signature = "keyId=\"${__P(tppDistinguishedName)}\",
                    algorithm=\"rsa-sha256\",
                    headers=\"" + headers + "\",
                    signature=" + "\"" + encryptedSignature + "\"";
```

Examples

PIS signature creation

Assume a TPP needs to include a signature in the following Request:

```
POST https://api.testbank.com/v1/payments/sepa-credit-transfers
Content-Type: application/json
X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address: 192.168.8.78
PSU-ID: PSU-1234
PSU-User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
tpp-redirect-uri: https%3A%2F%2FshortURI_Cchallenge_Mmethod="S256"
Date: Sun, 06 Aug 2017 15:02:37 GMT
{
   "instructedAmount": {"currency": "EUR", "amount": "123"},
   "debtorAccount": {"iban": "DE23100100101234567 89"},
   "creditor": {"name": "Merchant123"},
   "creditorAccount": {"iban": "DE23100120020123456789"},
   "remittanceInformationUnstructured": "Ref Number Merchant"
}
```

So the body would encode to the following String in Base64:

```
eyAgICANCiAgICJpbnN0cnVjdGVkQW1vdW50IjogeyJjdXJyZW5jeSI6ICJFVVIiLCAiYW1vdW50IjogIjEyMyJ9
LA0
KICAgImRlYnRvckFjY291bnQiOiB7ImliYW4iOiAiREUyMzEwMDEwMDEwMTIzNDU2Nzg5In0sDQogICAiY3JlZGl
0b3
IiOiB7Im5hbWUiOiAiTWVyY2hhbnQxMjMifSwNCiAgICJjcmVkaXRvckFjY291bnQiOiB7ImliYW4iOiAiREUyMz
EwM DEyMDAyMDEyMzQ1Njc4OSJ9LA0KICAg
InJlbWl0dGFuY2VJbmZvcm1hdGlvblVuc3RydWN0dXJlZCI6ICJSZWYgTnVtYmVyIE1lcmNoYW50Ig0KfQ==
```

And SHA-256 of the request body is:

```
iXhCYo105ae/y5v/UJkQWuBe1I+mdKG0JxwU35vwsgo=
```

So using signature algorithm rsa-sha256 the signed request of the TPP will be:

```
POST https://api.testbank.com/v1/payments/sepa-credit-transfers
Content-Type: application/json
X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address: 192.168.8.78
PSU-ID: PSU-1234
PSU-User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
tpp-redirect-uri: https%3A%2F%2FshortURI_Cchallenge_Mmethod="S256"
Date: Sun, 06 Aug 2017 15:02:37 GMT
Digest: SHA-256=ZuYiOtZkVxhjWmwTO5lOpsPevUNMezvk6dfb6fVhebM=
Signature: keyId="SN=9FA1,CA=CN=D-TRUST%20CA%202-1%202015,O=D-Trust%20GmbH,C=DE",algorithm="rsa-sha256",
headers="digest x-request-id",
signature="Base64(RSA-SHA256(signing string))"
TPP-Signature-Certificate: TPP's_eIDAS_Certificate
{
"instructedAmount": {"currency": "EUR", "amount": "123"},
"debtorAccount": { "iban": "DE23100100101234567 89"},
"creditor": { "name": "Merchant123"},
"creditorAccount": {"iban": "DE23100120020123456789"},
"remittanceInformationUnstructured": "Ref Number Merchant"
}
```

Where signing string is:

```
digest: SHA-256=iXhCYo105ae/y5v/UJkQWuBe1I+mdKG0JxwU35vwsgo= x-request-id: 99391c7e-
ad88-49ec-a2ad-99ddcb1f7721
```

> **Note:** The header field names to be signed are denoted in small letters to clarify that the digest will use small letters for normalisation.

AIS example

Example of create consent in AIS without signed request:

```
POST https://<HOST>/berlingroup/v1/consents HTTP/1.1
Connection: keep-alive
x-request-id: 4e73b197-fa18-4d29-9c81-00538d4088ca
apiKey: 3d6e4c19-dca6-4c85-a507-3b9f98be9111
content-type: application/json
psu-ip-address: 192.168.0.6
Content-Length: 359
Host: <HOST>
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_161)
{
"validUntil" : "2019-08-02",
"frequencyPerDay" : 1,
"recurringIndicator" : false,
"combinedServiceIndicator" : false,
"access" : {
"accounts" : [ {
"iban" : "BE63999001487608"
} ],
"balances" : [ {
"iban" : "BE63999001487608"
} ],
"transactions" : [ {
"iban" : "BE63999001487608"
} ]
}
```

This request including signed request:

```
POST https:///berlingroup/v1/consents HTTP/1.1
Connection: keep-alive
x-request-id: 4e73b197-fa18-4d29-9c81-00538d4088ca
apiKey: 3d6e4c19-dca6-4c85-a507-3b9f98be9111
content-type: application/json
digest: SHA-256=z18RGk+oEX/b32Kkkh3tNMD4KvKhOhnnUinyNQRlLvY=
Signature: keyId=" SN=4145E06EFDA52DC2,CA=EMAILADDRESS=rootCA@root.com, OU=DxP,
 O=SBS,L=P ",algorithm="rsa-sha256", headers="digest x-request-id",
 signature="Base64(RSA-SHA256(signing string))"
psu-ip-address: 192.168.0.6
tpp-signature-certificate: -----BEGIN CERTIFICATE-----
MIIERzCCAy+gAwIBAgIIQUXgbv2lLcIwDQYJKoZIhvcNAQELBQAwaTELMAkGA1UE
BhMCRlIxDjAMBgNVBAgTBVBhcmlzMQ4wDAYDVQQHEwVQYXJpczEMMAoGA1UEChMD
U0JTMQwwCgYDVQQLEwNEeFAxHjAcBgkqhkiG9w0BCQEWD3Jvb3RDQUByb290LmNv
bTAeFw0xOTAyMTQxMzU4MDBaFw0yODEyMTkxODAwMDBaMIGSMQswCQYDVQQGEwJG
UjEOMAwGA1UECBMFUGFyaXMxDjAMBgNVBAcTBVBhcmlzMQwwCgYDVQQKEwNTQlMx
DDAKBgNVBAsTA0R4UDEMMAoGA1UEAxMDU0JTMR4wHAYJKoZIhvcNAQkBFg9yb290
Q0FAcm9vdC5jb20xGTAXBgNVBGETEFBTREJFLUFJUy0xMjM0NTYwggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQCyB1NOqlS7X/XOZyoMPXHhDG8c8VyVC6sC
M6dW2rTkQEr4fm57fi3bdPjMTBRFB+dRNcERyWUSWiE85LW9UWcMIORltL59utio
1dySGUV+DRKffLAoe2jyhq5iWsU6WvK3kVcqLddXMfgdSB+qgYhQqCK6mpOilhdG
BE6nFbF4M4+sTGL/OncW4eGqAIsk6VvCEjuXtLabeE0Ji2BSekQnYMguOqEXkwcg
TPkMw1topWz3VNAgN0aGAOMAN0piFBGayoLVGqoLd3eD4Xk/EzFkt3/XBYZ1TSaF
1jevtwrscrbnIrajfg1R36a5gJAUZpweT3RvpIVsYDBdCSUbCPsjAgMBAAGjgcgw
gcUwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUQHK9dnN045oBeO2Be73LDTJU
VlswCwYDVR0PBAQDAgEGMBQGA1UdIAQNMAswCQYHBACL7EABATA9BggrBgEFBQcB
AwQxMC8wLQYGBACBmCcCMCMwEzARBgcEAIGYJwEDDAZQU1BfQUkMBEF1dGgMBkZy
YW5jZTARBglghkgBhvhCAQEEBAMCAAcwHgYJYIZIAYb4QgENBBEWD3hjYSBjZXJO
aWZpY2F0ZTANBgkqhkiG9w0BAQsFAAOCAQEAgPKZSlIVmemajqUe290ecKQ3HLd4
KruOMEqpJoIOPRF9zIXe9RQb7qwbYpP0tweL4tA8eEdkwz7dxOwno/75GeCb5vVo
1GDbeOecgIyEMjEPyxoTtuo1qr07GIcVXmx/aOMLoZXAbKq1Tg2SU+876z1YlBs8
4n4hloJeWpkbKOTQkzfmTWW1AtLrBnU7KFhKdC1s5g/EyVz063lscSJNgO7/i+n2
LSvrOptLOogZ92EcuVB6t7/3tmp6lIptMxoTNwZee8mqDI7ogUxcZUxGpAJzG6Ub
2qmL9KvMFjKTMhfyuTG8xx+4EN+VsduIxe/7/0nM5zQFwEZvPX1CQizAzQ==
-----END CERTIFICATE-----
Content-Length: 359
Host: <HOST>
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_161)
{
      "validUntil" : "2019-08-02",
      "frequencyPerDay" : 1,
      "recurringIndicator" : true,
      "combinedServiceIndicator" : false,
      "access" : {
            "accounts" : [ {
                  "iban" : "BE63999001487608"
            } ],
            "balances" : [ {
                  "iban" : "BE63999001487608"
            } ],
            "transactions" : [ {
                  "iban" : "BE63999001487608"
            } ]
```