OpenBanking eIDAS certificate handling

The purpose of this document is to describe the process of how Open Banking securely handles the eIDAS (electronic IDentification, Authentication and trust Services) certificate of its customers. In the Open Banking whitelabel integration, Klarna communicates with the ASPSP (Account Servicing Payment Service Provider) on behalf of the customer.

The customer's eIDAS certificate is used for the following purposes:

- Identification and authentication of the customer towards the ASPSP (QWAC (Qualified Website Authentication Certificate), transport layer).
- Encryption of the traffic between Klarna and ASPSP (QWAC, transport layer).
- Generating a digital signature of the payload to ensure the integrity and as an additional method for identification/authentication (QSealC (Qualified Electronic Seal Certificate), application layer).

As the eIDAS certificate offers the ability to prove the identity of the customer, it is considered as sensitive and highly confidential data and must be protected and secured.

Non-technical security measures

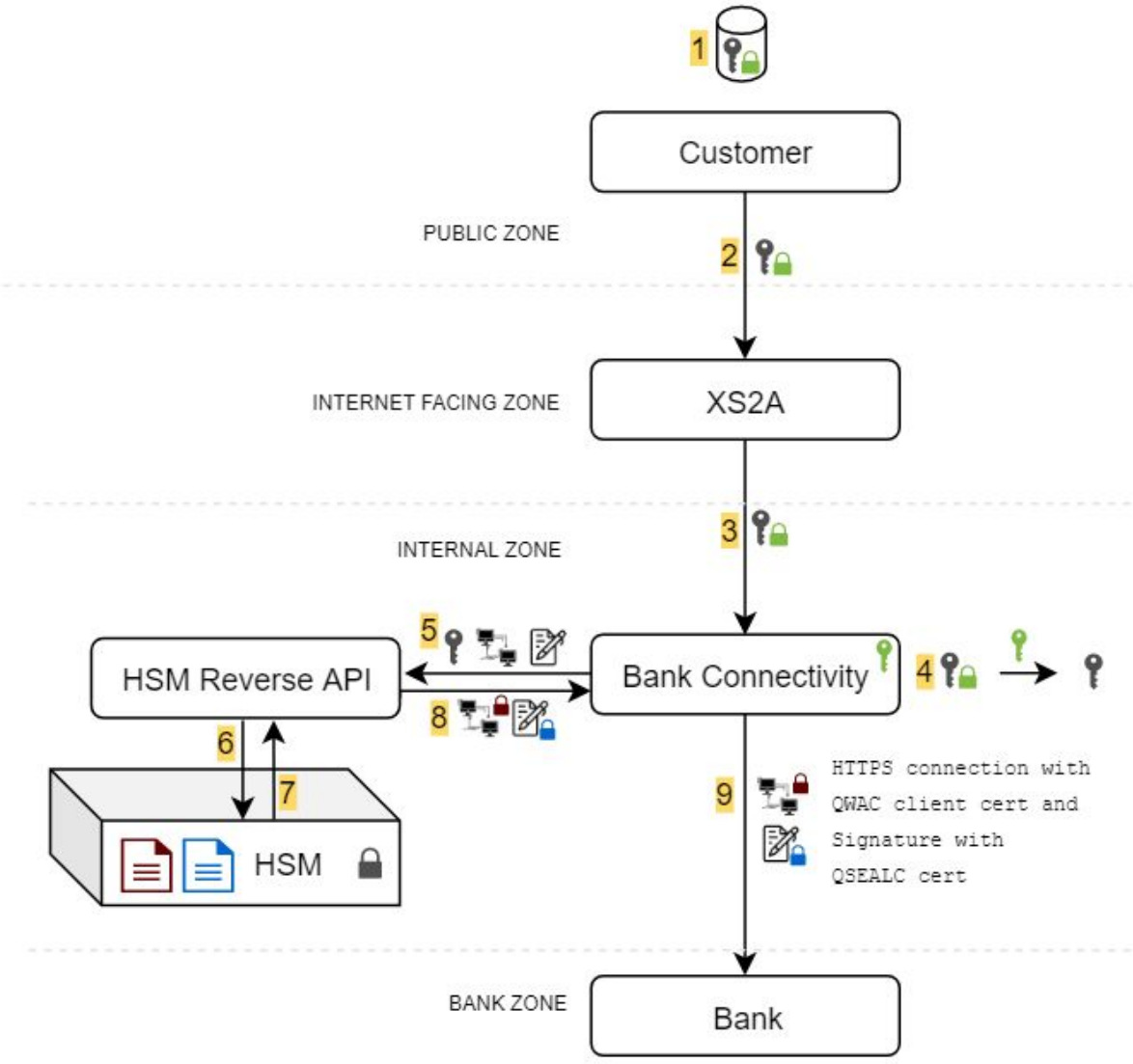
- In the contract between the customer and Klarna, it is clearly stated that it is an obligation of Klarna to only use the eIDAS certificates in the scope of the defined services of the agreement.
- The lifecycle of the certificates fully remains in the responsibility of the customer. The certificates used for the Open Banking integration can be revoked by the customer at any time.
- The customer can (and should) use a separate set of eIDAS certificates only for the Open Banking integration. This makes the certificate lifecycle independent and ensures business continuity of the other operational business of the customer in case of a revocation and creates a direct mapping from the specific eIDAS certificates to the Open Banking use-case. The EBA describes/recommends the same in one of their published letters:

However, in the specific cases where PSPs provide services through agents or EEA branches, or where they have outsourced to technical service providers some of the activities related to access to the online accounts held within an ASPSP, the EBA hereby clarifies that CAs should encourage these PSPs to consider using multiple certificates simultaneously: one per agent, EEA branch or technical service provider. This should ensure business continuity and better risk management of these PSPs because the legitimacy of one certificate would not be affected by the revocation of any other. PSPs remain fully responsible and liable for the acts of their agents and outsource providers as well as for the revocation and updating of the eIDAS certificates used by them.

Technical security measures

Process

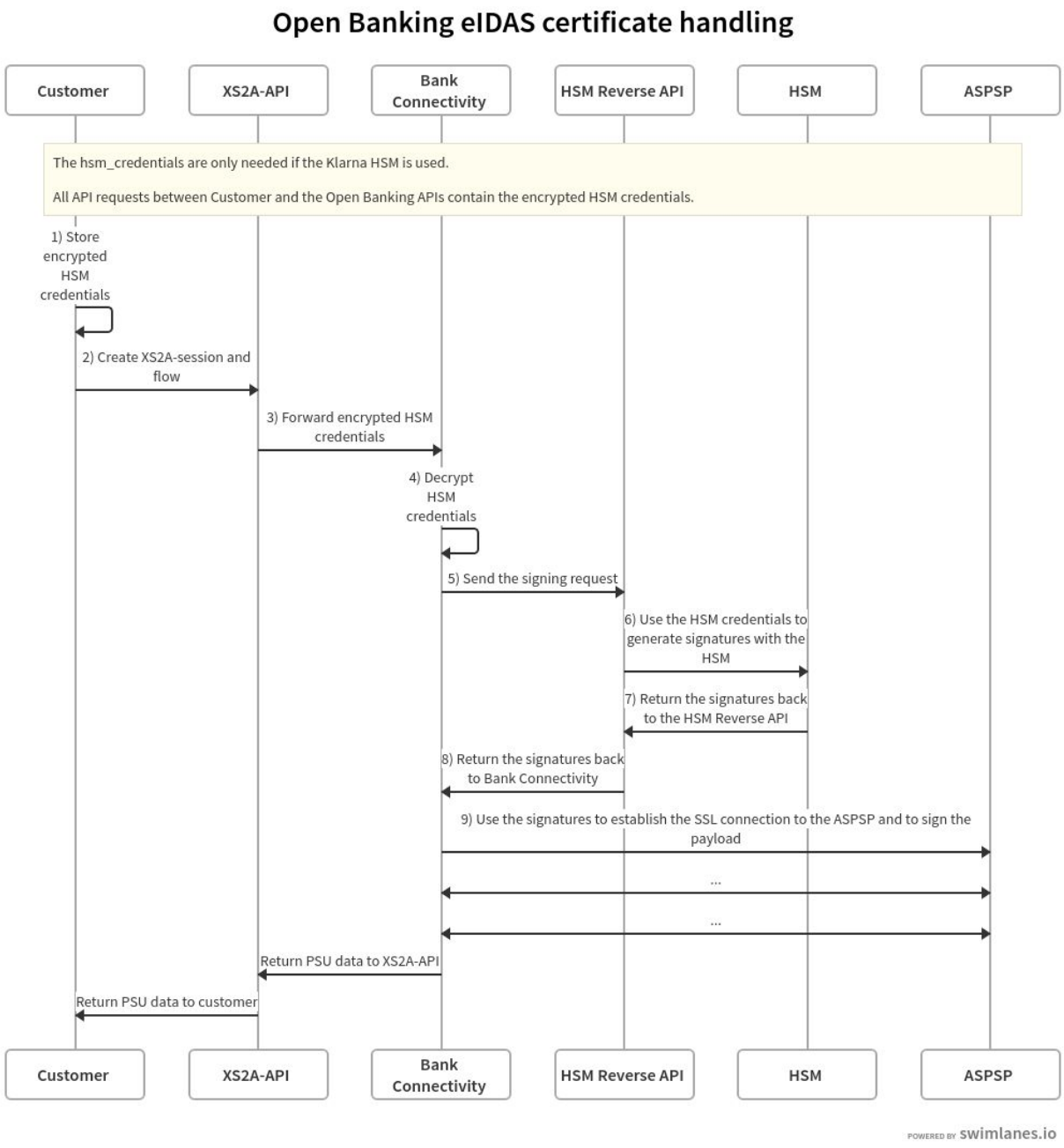
The following approach is assessed as the best and most secure:



1. The customer uses the encrypted HSM credentials to prepare the request to the Open Banking API (only needed if the Klarna HSM is used).
2. The customer executes an API request towards the Open Banking API and includes the encrypted HSM credentials.

3. The Open Banking API request is forwarded to the internal Bank Connectivity service.
4. The Bank Connectivity service decrypts the encrypted HSM credentials.
5. The Bank Connectivity service sends the signing request to the HSM Reverse API.
6. The HSM Reverse API uses the decrypted HSM credentials to access the HSM. The HSM signs the SSL handshake with the QWAC client certificate (private key) and the payload of the request to the ASPSP with the QSealC certificate (private key).
7. The HSM returns both signatures to the HSM Reverse API.
8. The HSM Reverse API sends the signatures back to the Bank Connectivity service.
9. The Bank Connectivity service uses the signatures to establish the SSL connection with the ASPSP and to add the signature to the payload of the request.

Swimlanes



Security measures explained

1. The eIDAS certificate is never transferred over the internet for the operational business.
2. No one can access the eIDAS certificate private keys.
3. The eIDAS certificate private keys are only stored in a FIPS-140-2 certified HSM.

4. The credentials for the HSM are only included by the customer for on-going requests, which ensures that Klarna can only use the credentials for the intended purpose and they are not available anymore for Klarna after the request is finished.
5. It is not possible for an employee to login to the Bank Connectivity instance (immutable infrastructure principle).
6. The application-code of the Bank Connectivity service, which can use the HSM Reverse API and the HSM to generate signatures for the purposes mentioned above, is controlled via a source control management (SCM) and changes to the code must be reviewed and approved by at least two other employees. This prevents misuse of the signatures for anything else except the agreed use-case.
7. (Klarna HSM only) The access credentials to use the eIDAS certificate on the HSM for generating signatures are never stored at Klarna.
8. (Customer HSM only) Klarna provides as many as possible fields to the HSM Reverse API to ensure that the customer can verify the payload that must be signed.
    a. The XS2A API session_id is included in all signing requests to correlate them to the Open Banking session and to verify that the session was started by the customer before.
    b. If the payload for signing contains a hash and plain-text data, the original payload and the hash_algorithm is provided too. This enables the validation of the real payload as well as the verification of the generated hash of the signing payload.

Using the Klarna HSM

There are three options available to upload the eIDAS private keys to Klarna's HSM.

- Send Klarna the private key, public key and the cert-file of the eIDAS certificates attached to an encrypted email, preferable via KeePass or GnuPG with symmetric passphrase ("gpg -c FILE"). Please do not include the password in this email, use a separate communication channel for it.
- Klarna generates the private key as "non-extractable" directly in the HSM. No one will and can ever see the key, but there is also no way to share it with the customer later, because it is fully protected by the HSM. Klarna creates a CSR (certificate signing request) based on the private key and the customer can use the CSR to request the eIDAS certificates. After the QWAC and QSeal certificates are issued, send them to Klarna via email, there is no need to highly protect them because only public information is contained.
- Take the opportunity and visit one of the Klarna offices :-). You can upload the private key to the HSM yourself, without sending it over any communication channel.

Using a customer owned/operated HSM

Klarna supports HSMs that are owned/operated by the customer. This enables the customer to fully control the eIDAS certificate private keys in its own HSM.

Requirements

- A secure connection from Klarna to the customer's HSM Reverse API service.
- Implementation of Klarna's "HSM Reverse API" (see below).
- The HSM must support signing of payloads with different algorithms that are typically used for the QWAC and QSealC certificates.

HSM Reverse API

Klarna uses a HSM Reverse API to integrate with customer owned/operated HSMs. This helps to secure the connection between Klarna and the customer's HSM because Klarna does not need direct access to the HSM and it avoids fragmentation in Klarna's service because the HSM Reverse API decapsulates the complex HSM APIs to one standardized API.

Sometimes the ASPSPs only sign the hash of the payload. To support the verification of the hash, Klarna will add the "digest_hash", "digest_hash_algorithm" and the "digest_payload" to the request too. In this case the value of "digest_hash" is one part of the "payload", but it is very likely that additional data is needed for the signature (see example below).

API definition

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id": "XS2A-API session_id to correlate this request to an existing session.",

  "alias": "Identifier for the key which must be used to sign the payload.",
  "algorithm": "Algorithm to use for signing.",
  "payload": "Base64 encoded payload that must be signed after decoding it.",
  "tls_client_auth": "Signature is part of the tls_client_auth for an HTTPS tls-handshake.",

  "digest_hash": "Hash of digest_payload generated with digest_hash_algorithm.",
  "digest_hash_algorithm": "The hash algorithm used to generate the digest_hash from digest_payload.",
  "digest_payload": "Base64 encoded input for the digest_hash."
}

Response:
{
  "signature": "Signature generated by the HSM, encoded as Base64."
}
```

API example data-types

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id": "UUID",

  "alias": "klarna-qwac-2019-07-01",
  "algorithm": "SHA256_RSA",
  "payload": "base64-string",
  "tls_client_auth": true/false,

  "digest_hash": ?"base64-string",
  "digest_hash_algorithm": ?"SHA256",
  "digest_payload": ?"base64-string"
}

Response:
{
  "signature": "base64-string"
}
```

API example AIS / consent

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qseal-2019-07-01",
  "algorithm" : "SHA256_RSA",
  "payload" :
"KHJlcXVlc3QtdGFyZ2V0KTogcG9zdCAvb2F1dGgyL3Rva2VuCmRhdGU6IFdlZCwgMzEgSnVsIDIwMTkgMTU6MTI6MjYgR01UCmRpZ2
VzdDogU0hBLTI1Nj13MG15bXVMOGFDcmJKbW1hYnMxcHl0Wmhvbjhs UXVjVHVKTVV0dUtyK3V3PQp4LWluZy1yZXFpZDogNjYwOTBlN
zEtYmQ1Yi00NGU2LTgwOTgtM2ZlYzU1NjhmZTVj",
  "tls_client_auth" : false,

  "digest_hash" : "w0mymuL8aCrbJmmabs1pytZhon8lQucTuJMUtuKr+uw=",
  "digest_hash_algorithm" : "SHA256",
  "digest_payload" : "Z3JhbnRfdHlwZT1jbGllbnRfY3JlZGVudGlhbHM="
}

Response:
{
  "signature": "SIGNATURE"
}
```

Example with the base-64 decoded values:
(the base64-decoded strings may contain additional linebreaks for better readability, for testing use the original base64-encoded values)

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qseal-2019-07-01",
  "algorithm" : "SHA256_RSA",
  "payload" : "(request-target): post /oauth2/token
date: Wed, 31 Jul 2019 15:12:26 GMT
digest: SHA-256=w0mymuL8aCrbJmmabs1pytZhon8lQucTuJMUtuKr+uw=
x-ing-reqid: 66090e71-bd5b-44e6-8098-3fec5568fe5c",
  "tls_client_auth" : false,

  "digest_hash" : "w0mymuL8aCrbJmmabs1pytZhon8lQucTuJMUtuKr+uw=",
  "digest_hash_algorithm" : "SHA256",
  "digest_payload" : "grant_type=client_credentials"
}

Response:
{
  "signature": "SIGNATURE"
}
```

API example PIS

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qseal-2019-07-01",
  "algorithm" : "SHA256_RSA",
  "payload" :
"ZGlnZXN0OiBTSEEtMjU2PTdPaCs1UG9hSFNEUXphYnkxTGZYUEZjTis1bFQvVXJzaWNKaDlBbERqK3c9CngtcmVxdWVzdC1pZDogZj
JlNGIwYjUtODUyNC00NTgzLWFkOWUtMmI0ZTkxNGMxNTMzCnBzdS1pZDogVlJLMTIzNDU2Nzg5ME9QVApkYXRlOiBUaHUsIDEgQXVnI
DIwMTkgMDg6MTg6MjggR01U",
  "tls_client_auth" : false,

  "digest_hash" : "7Oh+5PoaHSDQzaby1LfXPFcN+5lT/UrsicJh9AlDj+w=",
  "digest_hash_algorithm" : "SHA256",
  "digest_payload" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/PjxEb2N1bWVudCB4bWxucz0idXJuOm
lzbzpzdGQ6aXNvOjIwMDIyOnRlY2g6eHNkOnBhaW4uMDAxLjAwMS4wMyI+PENzdG1yQ2R0VHJmSW5pdG4+PEdycEhkcj48TXNnSWQ+M
TY2NDgxODUzMTAxOC01ODMwMTcwNTk8L01zZ0lkPjxDcmVEdFRtPjIwMTktMDgtMDFUMTA6MTg6MjcuOTM2KzAyOjAwPC9DcmVEdFRt
PjxOYk9mVHhzPjE8L05iT2ZUeHM+PEN0cmxTdW0+MTIzNDwvQ3RybFN1bT48SW5pdGdQdHkvPjwvR3JwSGRyPjxQbXRJbmY+PFBtdEl
uZklkPjE2NjQ4MjU1NDY1MzMtOTA5NzA2NjY8L1BtdEluZklkPjxQbXRNdGQ+VFJGPC9QbXRNdGQ+PE5iT2ZUeHM+MTwvTmJPZlR4cz
48Q3RybFN1bT4xMjM0PC9DdHJsU3VtPjxQbXRUcEluZj48U3ZjTHZsPjxDZD5TRVBBPC9DZD48L1N2Y0x2bD48L1BtdFRwSW5mPjxSZ
XFkRXhjdG5EdD4yMDE5LTA4LTAxKzAyOjAwPC9SZXFkRXhjdG5EdD48RGJ0ci8+PERidHJBY2N0PjxJZD48SUJBTj5ERTM5NDk5OTk5
NjAwMDAwMDA1MTExPC9JQkFOPjwvSWQ+PC9EYnRyQWNjdD48RGJ0ckFndD48RmluSW5zdG5JZD48T3Rocj48SWQ+Tk9UUFJPVklERUQ
8L0lkPjwvT3Rocj48L0Zpbkluc3RuSWQ+PC9EYnRyQWd0PjxDaHJnQnI+U0xFVjwvQ2hyZ0JyPjxDZHRUcmZUeEluZj48UG10SWQ+PE
VuZFRvRW5kSWQ+Tk9UUFJPVklERUQ8L0VuZFRvRW5kSWQ+PC9QbXRJZD48QW10PjxJbnN0ZEFtdCBDY3k9IkVVUiI+MTIzNDwvSW5zd
GRBbXQ+PC9BbXQ+PENkdHIvPjxDZHRyQWNjdD48SWQ+PElCQU4+REUwMjUwMDEwNTE3Mjg1NzY3MzMxNTwvSUJBTj48L0lkPjwvQ2R0
ckFjY3Q+PFJtdEluZj48VXN0cmQ+dGhpcyBpcyBhIHRlc3QgcHVycG9zZSB0ZXh0PC9Vc3RyZD48L1JtdEluZj48L0NkdFRyZlR4SW5
mPjwvUG10SW5mPjwvQ3N0bXJDZHRUcmZJbml0bj48L0RvY3VtZW50Pg==",
}
```

```
Response:
{
  "signature": "SIGNATURE"
}
```

Example with the base-64 decoded values:
(the base64-decoded strings may contain additional linebreaks for better readability, for testing, use the original base64-encoded values)

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qseal-2019-07-01",
  "algorithm" : "SHA256_RSA",
  "payload" : "digest: SHA-256=7Oh+5PoaHSDQzaby1LfXPFcN+5lT/UrsicJh9AlDj+w=\nx-request-id:
f2e4b0b5-8524-4583-ad9e-2b4e914c1533\npsu-id: VRK1234567890OPT\ndate: Thu, 1 Aug 2019 08:18:28 GMT",
  "tls_client_auth" : false,

  "digest_hash" : "7Oh+5PoaHSDQzaby1LfXPFcN+5lT/UrsicJh9AlDj+w=",
  "digest_hash_algorithm" : "SHA256",
  "digest_payload" : "<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
        <CstmrCdtTrfInitn>
                <GrpHdr>
                        <MsgId>1664818531018-583017059</MsgId>
                        <CreDtTm>2019-08-01T10:18:27.936+02:00</CreDtTm>
                        <NbOfTxs>1</NbOfTxs>
                        <CtrlSum>1234</CtrlSum>
                        <InitgPty/>
                </GrpHdr>
                <PmtInf>
                        <PmtInfId>1664825546533-90970666</PmtInfId>
                        <PmtMtd>TRF</PmtMtd>
                        <NbOfTxs>1</NbOfTxs>
                        <CtrlSum>1234</CtrlSum>
                        <PmtTpInf>
                                <SvcLvl>
                                        <Cd>SEPA</Cd>
                                </SvcLvl>
                        </PmtTpInf>
                        <ReqdExctnDt>2019-08-01+02:00</ReqdExctnDt>
                        <Dbtr/>
                        <DbtrAcct>
                                <Id>
                                        <IBAN>DE39499999600000005111</IBAN>
                                </Id>
                        </DbtrAcct>
                        <DbtrAgt>
                                <FinInstnId>
                                        <Othr>
                                                <Id>NOTPROVIDED</Id>
                                        </Othr>
                                </FinInstnId>
```

```
                        </DbtrAgt>
                        <ChrgBr>SLEV</ChrgBr>
                        <CdtTrfTxInf>
                                <PmtId>
                                        <EndToEndId>NOTPROVIDED</EndToEndId>
                                </PmtId>
                                <Amt>
                                        <InstdAmt Ccy="EUR">1234</InstdAmt>
                                </Amt>
                                <Cdtr/>
                                <CdtrAcct>
                                        <Id>
                                                <IBAN>DE02500105172857673315</IBAN>
                                        </Id>
                                </CdtrAcct>
                                <RmtInf>
                                        <Ustrd>this is a test purpose text</Ustrd>
                                </RmtInf>
                        </CdtTrfTxInf>
                </PmtInf>
        </CstmrCdtTrfInitn>
</Document>"
}

Response:
{
  "signature": "SIGNATURE"
}
```

API example TLS handshake of an HTTPS connection

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qwac-2019-07-31",
  "algorithm" : "SHA256_RSA",
  "payload" :
"AQABCQMDPona3JiNFU3YNiOfJHwTMkny0KGYnoL1Hwf6uxM8oMwAABTALMArwDAAncAvAJzAFAA1wBMALwEAAMwAAAAYABYAABNhcG
kuc2FuZGJveC5pbmcuY29tAAUABQEAAAAAAAoAFgAUABcAGAAZAAkACgALAAwADQAOABYACwACAQAADQAoACYEAwUDBgMIBAgFCAYIC
QgKCAsEAQUBBgEEAgMDAwEDAgIDAgECAgAyACgAJgQDBQMGAwgECAUIBggJCAoICwQBBQEGAQQCAwMDAQMCAgMCAQICABAADgAMAmgy
CGh0dHAvMS4xABEACQAHAgAEAAAAAAAAXAAAAKwADAgMD/wEAAQACAABcAwMSmypq9gMkUEt5sHasxL/nqDkvh+tToHXypQxZqdKOeyC
FIf5TiVSip83JoP1QfKoKeFQFtbFYa+DJqtVX4cy73sAwAAAU/wEAAQAAEAAFAAMCaDIACwACAQALABHhABHeAAehMIIHnTCCBoWgAw
IBAgIRAPdBDZrr/hdgAAAAAFTOXBcwDQYJKoZIhvcNAQELBQAwgboxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1FbnRydXN0LCBJbmMuM
SgwJgYDVQQLEx9TZWUgd3d3LmVudHJ1c3QubmV0L2xlZ2FsLXRlcm1zMTkwNwYDVQQLEzAoYykgMjAxNCBFbnRydXN0LCBJbmMuIC0g
Zm9yIGF1dGhvcml6ZWQgdXNlIG9ubHkxLjAsBgNVBAMTJUVudHJ1c3QgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkgLSBMMU0wHhcNMTg
wNDA5MDgzMzQxWhcNMjAwMzMwMDkwMzM4WjCBtzELMAkGA1UEBhMCTkwxFjAUBgNVBAgTDU5vb3JkLUhvbGxhbmQxEjAQBgNVBAcTCU
Ftc3RlcmRhbTETMBEGCysGAQQBgjc8AgEDEwJOTDEXMBUGA1UEChMOSU5HIEdyb2VwIE4uVi4xHTAbBgNVBA8TFFByaXZhdGUgT3JnY
W5pemF0aW9uMREwDwYDVQQFEwgzMzIzMTA3MzEcMBoGA1UEAxMTYXBpLnNhbmRib3guaW5nLmNvbTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAI7BQzwLSCREGdjsAdszjoGM4iPtAuDRVyn7FbM9OvGX7h0ilVcTYDduOLDtYwz7HS9SJvAn6AX57X2eytMGugp
Li3/WhV9CuZJt+UXWORI2OJCqmtA7kqFZz3lEqn0s9gvpdy01gmbVyCKqjToyO53aCS3hSAAktHNDeiuzrdxYNhjTV45L5lRmgeTGC+
XS+1/A0xzzFdz9iqFvHtYLdh0GaHySQyag8wr4wMK/p6wDF5/nrWnG+RdS723hKNef39L3dW4XuC5vivo0G5hqAHpjdJqPOAKrj/gwq
FogXRnW44Rb0hoormrYY0KBVmEOyIQ9ofn3ld+KDVa2lb+wL68CAwEAAaOCA50wggOZMDcGA1UdEQQwMC6CE2FwaS5zYW5kYm94Lmlu
Zy5jb22CF3d3dy5hcGkuc2FuZGJveC5pbmcuY29tMIIB9wYKKwYBBAHWeQIEAgSCAecEggHjAeEAdgBVgdTCFpA2AUrqC5tXPFPwwOQ
4eHAlCBcvo6odBxPTDAAAAWKppYtPAAAEAwBHMEUCIGxG8HYBHo3QaUVxmGuB3gkLTIOTeeeaPJPsQkx1j48xAiEAicXh0s24jFg7bN
DOoiNwv0l8Gea6YsVKfk2NH8rVjiMAdgC72d+8H4pxtZOUI5eqkntHOFeVCqtS6BqQlmQ2jh7RhQAAAWKppYsuAAAEAwBHMEUCIFZBt
```

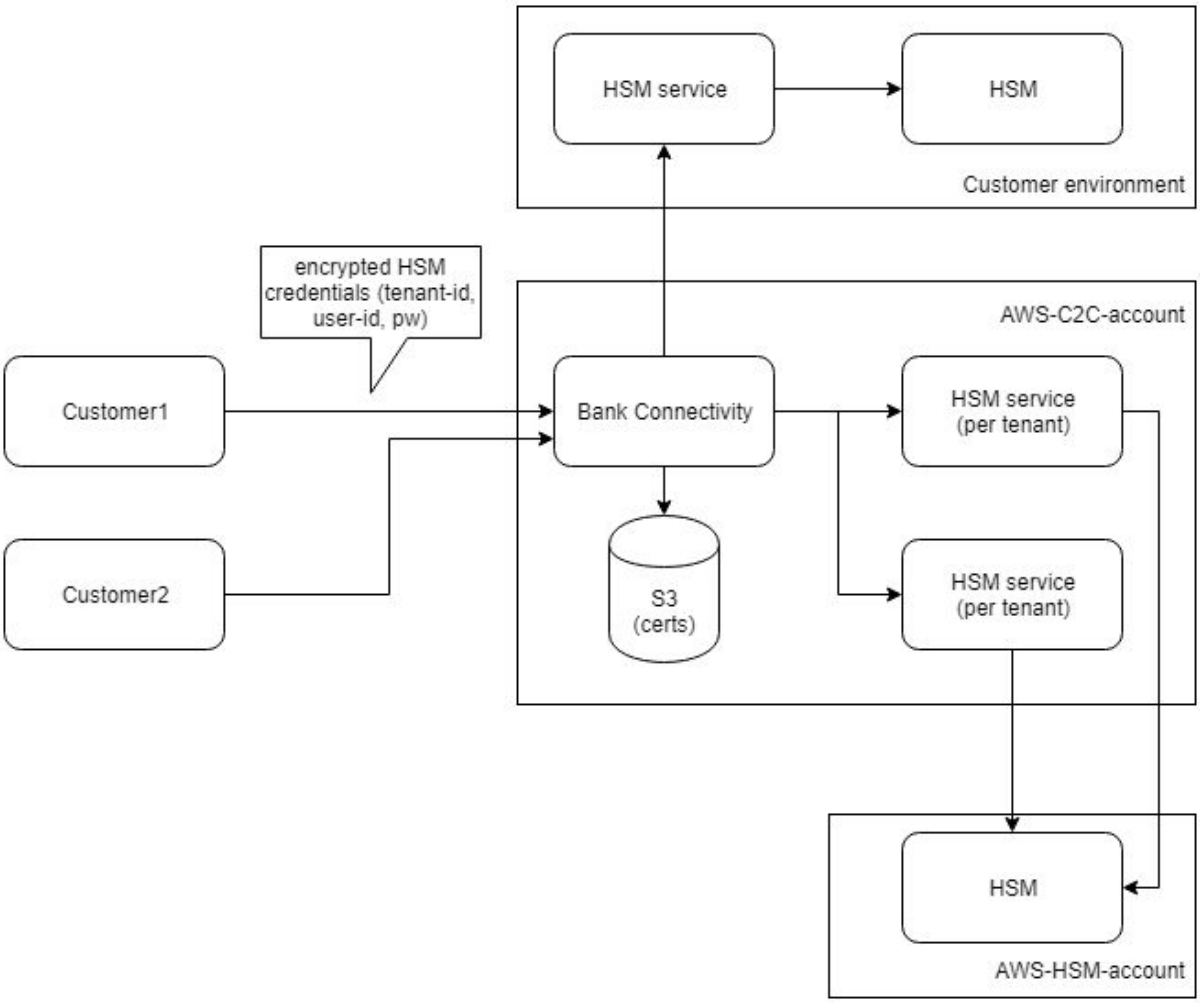
```
ZimeGn2G/1wXBnjcsX9JDPD309TGhsPFrqMZIOSAiEAlMrYJUCoh7OPGdLOdnkUz97EDxLQGNQ/ZaJRw8hWR7wAdgDd6x0reg1PpiCL
ga2BaHB+Lo6dAdVciI09EcTNtuy+zAAAAWKppYtrAAAEAwBHMEUCIA7xEUk22XJI/NgknXDUIUKhW3ZSf5XUu8hSdJRJxY7wAiEAsTL
WFBy8gJJmgMRGCb7svugfRSSwbc6jNK8a4UUaS08AdwCkuQmQtBhYFIe7E6LMZ3AKPDWYBPkb37jjd80OyA3cEAAAAWKppYsrAAAEAw
BIMEYCIQDgqxHCFeOC2TB0K9SsqMvb2TkW1OE6v1q+8hwBw0hHTwIhAL+KApMPj1AEcDRZMihEG2v8+Cv5TeKOqz0UdGzd/f3lMA4GA
1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwaAYIKwYBBQUHAQEEXDBaMCMGCCsGAQUFBzABhhdodHRw
Oi8vb2NzcC5lbnRydXN0Lm5ldDAzBggrBgEFBQcwAoYnaHR0cDovL2FpYS5lbnRydXN0Lm5ldC9sMW0tY2hhaW4yNTYuY2VyMDMGA1U
dHwQsMCowKKAmoCSGImh0dHA6Ly9jcmwuZW50cnVzdC5uZXQvbGV2ZWwxbS5jcmwwSgYDVR0gBEMwQTA2BgpghkgBhvpsCgECMCgwJg
YIKwYBBQUHAgEWGmh0dHA6Ly93d3cuZW50cnVzdC5uZXQvcnBhMAcGBWeBDAEBMB8GA1UdIwQYMBaAFMP30LUqMK2vDZEhcDlU3byJc
Mc6MB0GA1UdDgQWBBRDSsGReHRR/tW6pzyyqfHej9HS8DAJBgNVHRMEAjAAMA0GCSqGSIb3DQEBCwUAA4IBAQCEROcWo/BkoI0BjG8Z
RHvODauLXq6+RkLRqVrvVDRDuvX8cEX2bUYHcrCqGO+9pZCZeEnl710vc7jDql/Wqsn9+9bWUwMusm6QvJ0Jd1/hCY1lHP1/+mbajsi
ChpHCL3irHkjSOhRB4VEngs3BXyfg2uqKXJvVC8P/Ju8JDiwwTY/K7rSRwFh6zRJVGYCSXAWB6SFFfnIUKQVhWalj0J+5rlwMfQdqj7
rP5T10Z8IuMHYRyT75EeGie08yylXggke3MDZEvEqEujTYv4e9HuSrzbh7Pn3/fbs/8uvlvZd4w2ndxvn2bp5pFWUD4VqyzAeOt8aAu
1ioO0hM+DIU+vNhAAUxMIIFLTCCBBWgAwIBAgIMYaHn0gAAAABR02amMA0GCSqGSIb3DQEBCwUAMIG+MQswCQYDVQQGEwJVUzEWMBQG
A1UEChMNRW50cnVzdCwgSW5jLjEoMCYGA1UECxMfU2VlIHd3dy5lbnRydXN0Lm5ldC9sZWdhbC10ZXJtczE5MDcGA1UECxMwKGMpIDI
wMDkgRW50cnVzdCwgSW5jLiAtIGZvciBhdXRob3JpemVkIHVzZSBvbmx5MTIwMAYDVQQDEylFbnRydXN0IFJvb3QgQ2VydGlmaWNhdG
lvbiBBdXRob3JpdHkgLSBHMjAeFw0xNDEyMTUxNTI1MDNaFw0zMDEwMTUxNTU1MDNaMIG6MQswCQYDVQQGEwJVUzEWMBQGA1UEChMNR
W50cnVzdCwgSW5jLjEoMCYGA1UECxMfU2VlIHd3dy5lbnRydXN0Lm5ldC9sZWdhbC10ZXJtczE5MDcGA1UECxMwKGMpIDIwMTQgRW50
cnVzdCwgSW5jLiAtIGZvciBhdXRob3JpemVkIHVzZSBvbmx5MS4wLAYDVQQDEyVFbnRydXN0IEN1cnRpZmljYXRpb24gQXV0aG9yaXR
5IC0gTDFNMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0IHBOSPCsdHs91fdVSQ2kSAiSPf8ylIKsKs/M7WwhAf23056sP
uYIj0BrFb7cW2y7rmgD1J3q5iTvjOK64dex6qwymmPQwhqPyK/MzlG1ZTy4kwFItlngJHxBEoOm3yiydJs/TwJhL39axSagR3nioPvY
RZ1R5gTOw2QFpi/iuInMlOZmcP7lhw192LtjL1JcdJDQ6Gh4yEqI3CodT2ybEYGYW8YZ+QpfrI8wcVfCR5uRE7sIZlYFUj0VUgqtzS0
BeN8SYwAWN46lsw53GEzVc4qLj/RmWLoquY0djGqr3kplnjLgRSvadr7BLlZg0SqCU+01CwBnZuUMWstoc/B5QIDAQABo4IBKzCCASc
wDgYDVR0PAQH/BAQDAgEGMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDATASBgNVHRMBAf8ECDAGAQH/AgEAMDMGCCsGAQUFBw
EBBCcwJTAjBggrBgEFBQcwAYYXaHR0cDovL29jc3AuZW50cnVzdC5uZXQwMAYDVR0fBCkwJzAloCOgIYYfaHR0cDovL2NybC5lbnRyd
XN0Lm5ldC9nMmNhLmNybDA7BgNVHSAENDAyMDAGBFUdIAAwKDAmBggrBgEFBQcCARYaaHR0cDovL3d3dy5lbnRydXN0Lm5ldC9ycGEw
HQYDVR0OBBYEFMP30LUqMK2vDZEhcDlU3byJcMc6MB8GA1UdIwQYMBaAFGpyJnrQHu995ztpUdRsjZ+QEmarMA0GCSqGSIb3DQEBCwU
AA4IBAQC0h8eEIhopwKR47PVPG7SEl2937tTPWa+oQ5YvHVjepvMVWy7ZQ5xMQrkXFxGttLFBx2YMIoYFp7Qi+8VoaIqIMthx1hGOjl
J+Qgld2dnADizvRGsf2yS89byxqsGK5Wbb0CTz34mmi/5e0FC6m3UAyQhKS3Q/WFOv9rihbISYJnz8/DVRZZgeO2x28JkPxLkJ1YXYJ
Kd/KsLak0tkuHB8VCnTglTVz6WUwzOeTTRn4Dh2ZgCN0C/GqwmqcvrOLzWJ/MDtBgO334wlV/H77yiI2YIowAQPlIFpI+CRKMVe1QzX
1CA778n4wI+nQc1XRG5sZ2L+hN/nYNjvv9QiHg3nAAUDMIIE/zCCA+egAwIBAgIEUdNARDANBgkqhkiG9w0BAQsFADCBsDELMAkGA1U
EBhMCVVMxFjAUBgNVBAoTDUVudHJ1c3QsIEluYy4xOTA3BgNVBAsTMHd3dy5lbnRydXN0Lm5ldC9DUFMgaXMgaW5jb3Jwb3JhdGVkIG
J5IHJlZmVyZW5jZTEfMB0GA1UECxMWKGMpIDIwMDYgRW50cnVzdCwgSW5jLjEtMCsGA1UEAxMkRW50cnVzdCBSb290IENlcnRpZmljY
XRpb24gQXV0aG9yaXR5MB4XDTE0MDkyMjE3MTQ1N1oXDTI0MDkyMzAxMzE1M1owgb4xCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1FbnRy
dXN0LCBJbmMuMSgwJgYDVQQLEx9TZWUgd3d3LmVudHJ1c3QubmV0L2xlZ2FsLXRlcm1zMTkwNwYDVQQLEzAoYykgMjAwOSBFbnRydXN
0LCBJbmMuIC0gZm9yIGF1dGhvcml6ZWQgdXNlIG9ubHkxMjAwBgNVBAMTKUVudHJ1c3QgUm9vdCBDZXJ0aWZpY2F0aW9uIEF1dGhvcm
l0eSAtIEcyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuoS2ctueDGvimekwAad26jK4lUEaydphTlhyz/72gnm/c2EGC
qUn2LNf00VOHHLWTjLycooP94MZ0GqAgABFHrDH55q/ElcnHKNoLwqHvWprDl5l8xx31dSFjXAhtLMy54ui1YY5ArG40kfO5MlJxDun
3vtUfVe+8OhuwnmyOgtV4lCYFjITXC94VsHClLPyWuQnmp8k18bs0JslguPMwsRFxYyXegZrKhGfqQpuSDtv29QRGUL3jwe/9VNfnD7
0FyzmaaxOMkxid+q36OW7NLwZi66cUee3frVTsTMi5W3PcDwa+uKbZ7aD9I2lr2JMTeBYrGQ0EgP4to2UYySkcQIDAQABo4IBDzCCAQ
swDgYDVR0PAQH/BAQDAgEGMBIGA1UdEwEB/wQIMAYBAf8CAQEwMwYIKwYBBQUHAQEEJzAlMCMGCCsGAQUFBzABhhdodHRwOi8vb2Nzc
C5lbnRydXN0Lm5ldDAzBgNVHR8ELDAqMCigJqAkhiJodHRwOi8vY3JsLmVudHJ1c3QubmV0L3Jvb3RjYTEuY3JsMDsGA1UdIAQ0MDIw
MAYEVR0gADAoMCYGCCsGAQUFBwIBFhpodHRwOi8vd3d3LmVudHJ1c3QubmV0L0NQUzAdBgNVHQ4EFgQUanImetAe733nO2lR1GyNn5A
SZqswHwYDVR0jBBgwFoAUaJDkZ6SmU4DHhmak8fdLQ/uEvW0wDQYJKoZIhvcNAQELBQADggEBAGkzg/woem99751V68U+ep11s8zDOD
bZNKIoaBjqHmnTvefQd9q4AINOSs9v0fHBIj905PeYSZ6btp7h25h3LVY0sag82f3Azce/BQPUAsXx5cbaCKUTx2IjEdFhMB1ghEXve
ajGJpOkt800uGnFE/aRs8lFc3a2kvZ2ClvhA0e36SlMkTIjN0qcNdh4/R0f5IOJJICtt/nP5F2l1HHEhVtwH9s/HAHrGkUmMRTMZb9n
3srMM2XlQZHXN75BGpad5oqXnafOrE6aPb0BoGrZTyIAi0TVaWJ7LuvMuueSfWlnPfy4fN5Bh9Bp6roKGHoalUOzeXEodm2h+1dK7E3
IDhAMAAFJAwAXQQQpcpIdtnX0OIlGD9TRjrIAmI5WWw3+T2to80GKz0Ew/BXUat5nxHXpAJuuwiT5JpHHyVEQLpbqml/s/i8j/9k3BA
EBAGwcVqbX1nRY45c191TiaJV6HEfIyVH0oTzPUCjALvtbiDIv55pbIjrehSP+9V51kiRJiexxDkusXYHMFDoAyU6Biv1hV/o9avJvE
f7jgIZ1DzCBGmqGO5q/hnnWD+WxsWN4GXBFNgen3GsNdBbh63weQhLosOtA8iuSX693b+fhsg8cs/geY4B2fM0vRBTVygclgKmWynYr
Oz5t6pkBIv3p7LH6oFUVdnZijzL+O+NK4nGkv+h25oirXszepW076PJEEXswPcd/4P0J7yI6o4/itCtSHMl5dT0tYEWDERsvZpe4X8z
K1EZCUXjXNvqrb6UQSs68spauCTeohygUSIgNAAAgAwECQAAYBAEEAgQDBQEFAgUDBgEGAgYDAgECAgIDAAAOAAAACwAD/QAD+gAD9z
CCA/MwggLboAMCAQICBEmWAtIwDQYJKoZIhvcNAQELBQAwcjEfMB0GA1UEAwwWQXBwQ2VydGlmaWNhdGVNZWFuc0FQSTEMMAoGA1UEC
wwDSU5HMQwwCgYDVQQKDANJTkcxEjAQBgNVBAcMCUFtc3RlcmRhbTESMBAGA1UECAwJQW1zdGVyZGFtMQswCQYDVQQGEwJOTDAeFw0x
OTAzMDQxMzU5MDhaFw0yMDAzMDQxNDU5MDhaMDwxGzAZBgNVBAsMEnNhbmRib3hfZWlkYXNfcXdhYzEdMBsGA1UEYQwUUFNETkwtU0J
YLTEyMzQ1MTIzNDUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCDS/VFSaWsUKxw+lBFH/7xfpRQQNtDiqpn4dS1cpRxZg
vjNnikFn9ROqjlILCYIC8FvkclN0kqzjNSihQKzUyyrbxoXTxw/dHfDIEOddGzJIO0X0Pi3R3hi4ruGeU+as+Dx3z2+TGZTpnYWlWwI
Vi2OKkyFoXPGTGYzXRXCFnCwnQN8Rr6W9lBVq4eEyxYAGcE84UumakPz/OS3FoEzz6ZGenGrvhJO+cA1ly1eySL7egsD8rnHButKAnN
szfozlp+/+VAuMmZtv6VIlrfyQHL/WQkaUeKqU5YukAOgyaryZfiiQEGk4Yr35FTguMjS7FsAsmVprEn79755NraSfN7AgMBAAGjgcY
wgcMwFQYDVR0fBA4wDDAKoAigBocEfwAAATAhBgNVHSMEGjAYoBYEFHBIu14AwPQm+MR00XjSxE7ftHJZMIGGBggrBgEFBQcBAwR6MH
gwCgYGBACORgEBDAAwEwYGBACORgEGMAkGBwQAjkYBBgMwVQYGBACBmCcCMEswOTARBgcEAIGYJwEDDAZQU1BfQUkwEQYHBACBmCcBA
```

```
QwGUFNQX0FTMBEGBwQAgZgnAQIMBlBTUF9QSQwGWC1XSU5HDAZOTC1YV0cwDQYJKoZIhvcNAQELBQADggEBAF9nHeVVePG5HPaNYv/3
9CxvCIb0ImqhWCRbARMmloyWK01SnGNcOmi/YRJwXyGPunF4hwA5KCPXCRknjhUcPXPOlsqntXEITK4uN6LNrL0ArjK/6fRxRbOhb3V
SWvrjVweFNsBAWPvjdAz3uuV4lu6CWJJ7WcaWtwHyKQ0tvsqOa7nljLAoQcQuSmzaSg7ddVDdeAZsZ8LvcsxlEJHXzhNMVssPVZstl+
aCVbAbBFGvlGeRlTi6t+m0DcwknKjtSQiu2UHx2ivPVXx6wsf8zcfiDQT64k6qQ9HM0AtvJbJ7QdiTUTJLGWUoqNbYsKsAksQrQ9QT4
l+SlHr0LHzsUwEQAABCQQTiIYordSIEpThe0mcyBBlEENuyfDCu/Q8KX3QZ2LBDVoEpBMxKmXZaN2mCd/ymLvty+f4w3wFA9SJUACBT
VjgU",
  "tls_client_auth" : true,

  "digest_hash" : null,
  "digest_hash_algorithm" : null,
  "digest_payload" : null
}

Response:
{
  "signature": "SIGNATURE"
}
```

Example with the base-64 decoded values:

```
POST /sign
Authentication+Encryption: PGP

Request:
{
  "session_id" : "175cnd9qoj7i9sh4ihf8ch8jrnc6th7t",

  "alias" : "klarna-qwac-2019-07-31",
  "algorithm" : "SHA256_RSA",
  "payload" : "binary data from tls handshake, usually server certificate",
  "tls_client_auth" : true,

  "digest_hash" : null,
  "digest_hash_algorithm" : null,
  "digest_payload" : null
}

Response:
{
  "signature": "SIGNATURE"
}
```

HSM setup


HSM service
HSM
Customer environment
encrypted HSM credentials (tenant-id, user-id, pw)
AWS-C2C-account
Customer1
Bank Connectivity
HSM service (per tenant)
Customer2
S3 (certs)
HSM service (per tenant)
HSM
AWS-HSM-account