A-Trust GmbH
Landstraßer Hauptstraße 1b
The Mall E02
A-1030 Wien

https://www.a-trust.at
E-Mail: office@a-trust.at

Tel: +43 (1) 713 21 51 - 0
Fax: +43 (1) 713 21 51 - 350

a.sign RK HSM

Developer Manual

Version: 1.1
Date:

Contents

Datum	Rev	Autor	Änderungen
10.12.2024	1.1	A. Kopp-Konopka	Aktualisierung Vorlage
18.09.2017	1.0	Patrick Hagelkruys	Erklärung REST URL
16.05.2017	0.9	Patrick Hagelkruys	Schnittstelle V2
22.02.2017	0.8	Patrick Hagelkruys	Verlinkung aktuelle RKSV Version
09.01.2017	0.7	Patrick Hagelkruys	Fehlermeldungen hinzugefügt
29.08.2016	0.6	Daniel Kovacic	Korrekturen
26.08.2016	0.5	Patrick Hagelkruys	Korrektur API Pfade in Beispielen Kapitel Authentifizierung
12.08.2016	0.4	Patrick Hagelkruys	API Ausstellen von Zertifikaten
20.07.2016	0.3	Patrick Hagelkruys	kleinere textuelle Änderungen
27.06.2016	0.2	Patrick Hagelkruys	Änderung Produktname
13.04.2016	0.1	Patrick Hagelkruys	Erste Version

Table 1: Dokumentenhistorie

1. Overview

1.1. Summary

This Document describes the REST-Interface of a.sign RK HSM.

The a.sign RK HSM is a signature server for creating digital signatures according to the Austrian cash register security regulation [Bun17]

1.2. Overview

The interface to a.sign RK HSM is implemented using a REST Interface (HTTP POST and HTTP GET). The *keylabel* has to be provided in the URL, which allows the signing key to be selected.

1.3. Keylabel

The *keylabel* is obtained when the certificate is ordered, where a free key has to be selected for which a certificate using TaxNumber, GLN or UID will be issued.

The section Inventory in the WebInterface of a.sign RK HSM lists all available keylabels. If a certificate has already been issued for a certificate, then the "Ordnungsbegriff" (eg. ATU12345678) is shown in the column next to it.

The keylabel has the following format: eg. `BEKUQDUIRNCOEYEDOSWFENGBCGWJGBYP`

1.4. Signature creation

This command is a POST with the data to be signed in JSON format as its content.

The provided data will first be hashed and then the ECDSA key is applied, according to [Jon15, chapter 3.1] the appropriate algorithm will be chosen. Currently a ECC P-256 key will be used, hence the hash function is SHA-256 which leads to JWS algorithm ES256.

The response contains the generated signature [Jon15, chapter 3.4] in the following format.

$$\begin{aligned} signature &= Base64\_url(\ R + \ S) \\ \text{where } R &= \text{first coordinate of the ECDSA point} \\ S &= \text{second coordinate of the ECDSA point} \end{aligned}$$

Request

```
POST /aSignRkHsm/V2/{keylabel}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 102

{
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 1: Signatur Request V2

Response

```
HTTP/1.1 200 OK
Content-Length: 130
Content-Type: application/json; charset=utf-8

{
"signature":"BC4jJ_fdAvBBln-y6h...egC7U",
}
```

Listing 2: Signatur Response V2

1.4.1. Signature creation, providing the hash value

Like the command in 1.4 instead of the Plaintext this command requires the hash value. Hence the hash has to be calculated on the client side.
The hash algorithm to be used is dependent on the signature algorithm (see chapter A.2) and Table [Jon15, chapter 3.1].

Request

```
POST /aSignRkHsm/V2/{keylabel}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 102

{
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 3: Signature Request (Hash) V2

Response

```
HTTP/1.1 200 OK
Content-Length: 130
Content-Type: application/json; charset=utf-8

{
"signature":"BC4jJ_fdAvBBln-y6h...egC7U",
}
```

Listing 4: Signature Response (Hash) V2

1.4.2. Signature creation, JWS

The request is a POST containing the JSON which contains the data to be signed.

The JWS Header has to be generated according to [Bun17, Anlage Ziffer 13], the content is built according to JWS Standard [Jon15]. The JWS structure including the signature will be returned.

Request

```
POST /aSignRkHsm/V2/{keylabel}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 87

{
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 5: Signature Request (JWS) V2

Response

```
HTTP/1.1 200 OK
Content-Length: 189
Content-Type: application/json; charset=utf-8

{
"result":"ey...J9.X1I...z0.an...adQ",
}
```

Listing 6: Signature Response (JWS) V2

1.5. Certificate information

Querying the certificate information can be accomplished using a GET Request.

Request

```
GET /aSignRkHsm/V2/{keylabel}/Certificate HTTP/1.1
Host: ...
```

Listing 7: Certificate Information Request V2

Response

```
HTTP/1.1 200 OK
Content-Length: 1540
Content-Type: application/json; charset=utf-8

{
"Signaturzertifikat":"MIIE...QA6o=",
"Zertifizierungsstellen":["MII...WSF"],
"Zertifikatsseriennummer":"963244432",
"ZertifikatsseriennummerHex":"3969F190",
"alg":"ES256"
}
```

Listing 8: Certificate Information Response V2

The first two lines of the response are in the required format for "Beleg Gruppe" (see RKSV - Anlage Detailspezifikation - Z6). The parameter `alg` is required for the payload of the signature.

1.6. ZDA Information

Querying the ZDA Information can be done using this GET Request.

Request

```
GET /aSignRkHsm/V2/{keylabel}/ZDA HTTP/1.1
Host: ...
```

Listing 9: ZDA Information Request V2

Response

```
HTTP/1.1 200 OK
Content-Length: 15
Content-Type: application/json; charset=utf-8

{
"zdaid":"AT1"
}
```

Listing 10: ZDA Information Response V2

1.7. Issuing a certificate on a.sign RK HSM

Issuing a new certificate can be done using a POST Request containing the "Ordnungsbegriff" and an email address.

The response contains the `keylabel` (see chapter 1.3) for the issued a.sign RK HSM certificate.

The Ordnungsbegriff (e.g.: UID-Nummer) has to be provided in two parts:

There are three different Categories (UID, GLN or tax number) provided via a single integer value (classification_key_type). The second part is the value itself.

- classification_key_type=0; UID-Nummer;
  z.B.: classification_key=ATU12345678
- classification_key_type=1; Global Location Number (GLN);
  z.B.: classification_key=5012345000008
- classification_key_type=2; Finanzamt- und Steuernummer;
  z.B.: classification_key=12345/1234

Furthermore an email address has to be provided, which is used for information purposes (e.g. when the certificates go out of life). This email can be the email of the final customer or the email of the A-Trust partner. It is recommeded to use a group mailbox which will remain valid after employees leave the company, rather than a personal email address.

Authentication with a.sign RK HSM is performed via the `Authorization` header in the request.

Here the request uses HTTPS (SSL/TLS) and the headers are to be added. See chapter 1.8 or [Wik16].

Request

```
POST /api/v1/Ausstellen HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46dGVzdDEyMzQ=
Host: ...
Content-Length: 211

{
"classification_key_type": 0,
"classification_key": "ATU00000000",
"email": "test@test.com"
}
```

Listing 11: Certificate Generation Request

Response

```
HTTP/1.1 200 OK
Content-Length: 15
Content-Type: application/json; charset=utf-8

{
 "keylabel":"JBLPSFXGKMZIMIWQGPJXUZFUSZMFKTUD"
}
```

Listing 12: Certificate Generation Response

1.8. Authentification

Authentication is required when issuing a new certificate. If a.sign RK HSM is used over the internet or other insecure networks, all other commands should also be sent via https.

When using authentification, all commands are sent using HTTPS (SSL/TLS) and the HTTP header `Authorization` needs to be added. Following is an example of a request using the `Authorization` Header.

```
GET /aSignRkHsm/V2/{keylabel}/Certificate HTTP/1.1
Authorization: Basic YWRtaW46dGVzdDEyMzQ=
Host: ...
```

Listing 13: Example Authorization Header

The values in the `Authorization` header consists of the value `Basic` and the Base64 encoded value of the username and passowrd separated by a colon as illustrated below.

```
string user = "admin"
string pwd = "pwd1234"

string token = Base64(user + ":" + pwd)
string header_value = "Basic␣" + token

webRequest.AddHeader("Authorization",header_value)
```

A detailed description of Basic Authentication can be found here [Wik16].

2. Test System

For testing purposes the following HSM can be used:

URL: http://hs-abnahme.a-trust.at/aSignRkHsm/V2/

keylabel: testkeylabel

3. Authentification

In the standard configuration, the REST API of a.sign RK HSM doesn't require authentication. This way all clients in the network can use the HSM for signing purposes. This configuration **is not recommended for a configuration where the HSM can be accessed via internet**.

The Administrator Guide [Hag16] provides information on how to configure a.sign RK HSM, so that all REST-API Calls require authentification.

Authentication must be turned on if using a.sign RK HSM over **internet or other insecure networks**.

In the same way a.sign RK HSM can be configured to require TLS (attention: self-signed certificate).

Chapter 1.8 describes the required changes to the api calls.

A. Interface version V1

A.1. Signature creation

This command is a POST with the data to be signed in JSON format as its content.

The provided data will first be hashed and then the ECDSA key is applied, according to [Jon15, chapter 3.1] the appropriate algorithm will be chosen. Currently a ECC P-256 key will be used, hence the hash function is SHA-256 which leads to JWS algorithm ES256.

The response contains the generated signature [Jon15, chapter 3.4] in the following format.

$$\begin{aligned} signature &= Base64(\ R + \ S) \\ \text{where } R &= \text{first coordinate of the ECDSA point} \\ S &= \text{second coordinate of the ECDSA point} \end{aligned}$$

Request

```
POST /aSignRkHsm/{keylabel}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 102

{
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 14: Signatur Request

Response

```
HTTP/1.1 200 OK
Content-Length: 130
Content-Type: application/json; charset=utf-8

{
"signature":"BC4jJ\/fdAvBBln+y6h...egC7U=",
}
```

Listing 15: Signatur Response

The response value `signature` is base64 encoded.

A.1.1. Signature creation, providing the hash value

Like the command in A.1 instead of the Plaintext this command requires the hash value. Hence the hash has to be calculated on the client side.

The hash algorithm to be used is dependent on the signature algorithm (see chapter A.2) and Table [Jon15, chapter 3.1].

Request

```
POST /aSignRkHsm/{keylabel}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 102

{
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 16: Signature Request (Hash)

Response

```
HTTP/1.1 200 OK
Content-Length: 130
Content-Type: application/json; charset=utf-8

{
"signature":"BC4jJ\/fdAvBBln+y6h...egC7U=",
}
```

Listing 17: Signature Response (Hash)

The response value `signature` is base64 encoded.

A.1.2. Signature creation, JWS

The request is a POST containing the JSON which contains the data to be signed.

The JWS Header has to be generated according to [Bun17, Anlage Ziffer 13], the content is built according to JWS Standard [Jon15]. The JWS structure including the signature will be returned.

Request

```
POST /aSignRkHsm/{keylabel}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 87

{
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 18: Signature Request (JWS)

Response

```
HTTP/1.1 200 OK
Content-Length: 189
Content-Type: application/json; charset=utf-8

{
"result":"ey...J9.X1I...z0.an...adQ",
}
```

Listing 19: Signature Response (JWS)

A.2. Certificate information

Querying the certificate information can be accomplished using a GET Request.

Request

```
GET /aSignRkHsm/{keylabel}/Certificate HTTP/1.1
Host: ...
```

Listing 20: Certificate Information Request

Response

```
HTTP/1.1 200 OK
Content-Length: 1540
Content-Type: application/json; charset=utf-8

{
"Signaturzertifikat":"MIIE...QA6o=",
"Zertifizierungsstellen":["MII...WSF"],
"Zertifikatsseriennummer":"963244432",
"ZertifikatsseriennummerHex":"3969F190",
"alg":"ES256"
}
```

Listing 21: Certificate Information Response

The first two lines of the response are in the required format for “Beleg Gruppe” (see RKSV - Anlage Detailspezifikation - Z6). The parameter `algo` is required for the payload of the signature.

A.3. ZDA Information

Querying the ZDA Information can be done using this GET Request.

Request

```
GET /aSignRkHsm/{keylabel}/ZDA HTTP/1.1
Host: ...
```

Listing 22: ZDA Information Request

Response

```
HTTP/1.1 200 OK
Content-Length: 15
Content-Type: application/json; charset=utf-8

{
"zdaid":"AT1"
}
```

Listing 23: ZDA Information Response

B. Return codes of the interface

HTTP 200 Success

HTTP 400 (error parse request body) invalid request body

HTTP 500 (error parse certificate) error loading certificate data

HTTP 400 (keylabel not found) keylabel not found

HTTP 500 general error

References

[Bun17] Bundesministers für Finanzen: *Verordnung des Bundesministers für Finanzen über die technischen Einzelheiten für Sicherheitseinrichtungen in den Registrierkassen und andere, der Datensicherheit dienende Maßnahmen (Registrierkassensicherheitsverordnung, RKSV)*, 2017. https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20009390&FassungVom=2017-04-01, visited on 2017-02-22.

[Hag16] Hagelkruys, Patrick: *a.sign RK HSM Administrator Handbuch*, 2016.

[Jon15] Jones, M.: *JSON Web Algorithms (JWA)*. RFC 7518, May 2015. https://tools.ietf.org/html/rfc7518, visited on 2015-11-25.

[Wik16] Wikipedia: *Basic access authentication — Wikipedia, The Free Encyclopedia)*, 2016. https://en.wikipedia.org/wiki/Basic_access_authentication, visited on 2016-08-26.