A-Trust Gesellschaft für Sicherheitssysteme
im elektronischen Datenverkehr GmbH
Landstraßer Hauptstraße 1b
The Mall E02
A-1030 Wien

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

a.sign RK HSM beginner/advanced/premium

Developer Manual

Version: 2.2
Datum: 13. Februar 2020

Inhaltsverzeichnis

Datum	Rev	Autor	Änderungen
13.02.2020	2.2	Patrick Hagelkruys	Fehler im Beispiel von Session Sign JWS behoben
18.09.2017	2.1	Patrick Hagelkruys	Erklärung REST URL
12.06.2017	2.0	Patrick Hagelkruys	Umbenennen a.sign RK HSM Basic/Advanced/Premium
09.06.2017	1.7	Patrick Hagelkruys	Umbenennen a.sign RK MOIBLE - a.sign RK ONLINE.
06.07.2016	1.6	Patrick Hagelkruys	Anpassung URLs in Beispielen. Entfernen der V1 Version der Schnittstelle aus der Dokumentation. Erweiterte Beschreibungen für Live-System.
04.07.2016	1.5	Patrick Hagelkruys	Live-System Daten Umbenennen von `algo` zu `alg` Link zu Aktivierungshilfe
12.05.2016	1.4	Patrick Hagelkruys	Produkt umbenannt
28.04.2016	1.3	Patrick Hagelkruys	Fehler in Dokumentation Session Login
24.02.2016	1.2	Patrick Hagelkruys	Fehler in Kapitel Überschrift
22.02.2016	1.1	Patrick Hagelkruys	Zertifikatsseriennummer in Hex-Format
26.01.2016	1.0	Patrick Hagelkruys	Schnittstellen Interface V2 Version 1 der Schnittstelle im Anhang Befehl zum Ändern des Passworts Weitere Konten für Testfälle
04.01.2016	0.9	Ramin Sabet Patrick Hagelkruys	Interner Review
23.12.2015	0.8	Patrick Hagelkruys	Erzeugen einer Session für schnellere Signatur aufrufe Signatur von Plaintext Signatur JWS Anwendungsfälle
10.12.2015	0.7	Patrick Hagelkruys	Erstellen eines Kundenkonto liefert jetzt auch die Zertifikatsdaten zurück Algorithmus aus Signatur entfernt und zu Zertifikatsinformationen hinzugefügt.
26.11.2015	0.6	Patrick Hagelkruys	Englische Version PHP Beispielcode
16.11.2015	0.5	Patrick Hagelkruys	Klarstellungen bei Signaturbefehl Eigener Befehl für Signatur von Hash
30.10.2015	0.4	Patrick Hagelkruys	Überarbeitung
21.10.2015	0.3	Patrick Hagelkruys	Erweiterung der Schnittstelle
19.10.2015	0.2	Patrick Hagelkruys	Update Fehlercodes
16.10.2015	0.1	Patrick Hagelkruys	Erste Version

Tabelle 1: Dokumentenhistorie

1. Überblick

1.1. Zusammenfassung

Ziel dieses Dokumentes ist die Beschreibung der Schnittstelle zur a.sign RK HSM in den Ausführungen Basic, Advanced und Premium. Diese drei Ausführungen unterscheiden sich in der Anzahl der Signaturen und der Geschwindigkeit. Weitere Informationen sind unter http://www.a-trust.at/registrierkasse zusammengetragen

Der a.sign RK HSM ist ein Signatur Server zur Erstellung von digitalen Signaturen wie diese in der österreichischen Registrierkassen Sicherheitsverodnung [?] gefordert werden.

2. Schnittstelle

2.1. Übersicht

Als Schnittstelle zur a.sign RK HSM wird eine REST Schnittstelle (HTTP POST und HTTP GET) verwendet. In der URL muss ein Benutzername angegeben werden.

2.2. Signaturbefehle mit/ohne Session

Für die Signaturbefehle sind zwei Varianten verfügbar.

In der einfachen Variante ohne Session wird der Benutzernamen und das Passwort im Signaturaufruf übergeben. Hierbei muss mit dem Benutzername und Passwort ein Symmetrischer Schlüssel generiert werden welches zirka ein halbe Sekunde in Anspruch nimmt.

Daher wurde eine zweiten Variante des Signaturaufrufs umgesetzt um die Geschwindigkeit der Abarbeitung zu erhöhen. Bei dieser zweite Variante ist zuerst ein Session Login notwendig, hierbei wird die Schlüsselableitung (zirka eine halbe Sekunde) durchgeführt. Dieser Session Login liefert zwei Datenfelder zurück (`sessionid` und `sessionkey`) welche für die nachfolgenden Signaturbefehle benötigt werden.

2.3. Signaturbefehle ohne Session

2.3.1. Signatur erstellen

Der Signaturaufruf besteht aus einem POST, dessen Inhalt ein JSON-Objekt mit Passwort und zu signierende Daten (Base64 codierter Plaintext) ist.

Bei diesem Aufruf wird auf die übergebenen Daten zuerst die Hashfunktion und anschließend der ECDSA Schlüssel angewandt, entsprechend [?, Kapitel 3.1] wird der für die Schlüssellänge passende Algorithmus ausgewählt. Derzeit ist ein ECC P-256 Schlüssel in Verwendung, daher ist als Hashfunktion ein SHA-256 zu verwenden und es ergibt sich der JWS Algorithmus ES256.

Die Antwort enthält die erstellte Signatur entsprechend [?, Kapitel 3.4] im folgendem Format.

$$signature = Base64\_url(\ R + \ S)$$
$$\text{wobei } R = \text{erste Koordinate des ECDSA Punktes}$$
$$S = \text{zweite Koordinate des ECDSA Punktes}$$

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 1: Signatur Request

Antwort

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

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

Listing 2: Signatur Response

2.3.2. Signatur erstellen, Übergabe des Hashwertes

Dieser Befehl ist ähnlich der Signaturerstellung im Kapitel 2.3.1, jedoch wird anstelle der Base64 codierten Plaintext Daten der Hashwert übergeben. Damit muss der entsprechende Hash am Client durchgeführt werden.
Welcher Hashalgorithmus verwendet werden muss, ergibt sich aus dem Signaturalgorithmus (siehe Kapitel 2.5) und der Tabelle aus [?, Kapitel 3.1].

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 3: Signatur Request (Hash)

Antwort

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

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

Listing 4: Signatur Response (Hash)

2.3.3. Signatur erstellen, Übergabe des Plaintext

Dieser Befehl ist ähnlich der Signaturerstellung im Kapitel 2.3.1, jedoch ohne der Base64 Kodierung der Plaintext-Daten.
Für die Zusammenstellung des Plaintext ist in Kapitel 6.3 ein Beispiel angegeben.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign/Plain HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 5: Signatur Request (Plain)

Antwort

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

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

Listing 6: Signatur Response (Plain)

2.3.4. Signatur erstellen, JWS

Der Signatur Aufruf besteht aus einem POST, dessen Inhalt ein JSON mit Passwort und zu signierenden Daten ist.

Bei diesem Aufruf wird der JWS Header entsprechend der [?, Anlage Z 13] erzeugt, die übergebenen Daten entsprechend dem JWS Standard [?] formatiert und die gesamte JWS Struktur inkl. Signatur zurückgeliefert.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 247

{
"password":"123456789",
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 7: Signatur Request (JWS)

Antwort

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

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

Listing 8: Signatur Response (JWS)

2.4. Signaturbefehle mit Session

2.4.1. Session Login

Der Session Login besteht aus einem PUT-Request, dessen Inhalt ein JSON mit Passwort ist. Als Antwort werden eine SessionId und ein SessionKey zurückgeliefert welche für die nachfolgenden Signaturbefehle benötigt werden.

Eine Session ist eine Stunde lang gültig. Mit jedem erfolgreichen Signaturbefehl erhöht sich die Gültigkeit um eine Stunde. Spätestens um Mitternacht wird eine Session beendet.

Anfrage

```
PUT /asignrkonline/v2/Session/{Benutzername} HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 30

{
"password":"123456789"
}
```

Listing 9: Session Login Request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 120
Content-Type: application/json; charset=utf-8

{
"sessionid":"ervhiklakgcmifzgeuwwfuttsalffovl",
"sessionkey":"ak3k39oVApkGfZ92FhcbFmL38zK6EMunu4ooqh3foGY="
}
```

Listing 10: Session Login Response

2.4.2. Session Logout

Dieser Befehl dient zum vorzeitigen Beenden einer Session.

Anfrage

```
DELETE /asignrkonline/v2/Session/{sessionid} HTTP/1.1
Host: ...
```

Listing 11: Session Logout Request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 16
Content-Type: application/json; charset=utf-8

{"result":true}
```

Listing 12: Session Logout Response

2.4.3. Signatur erstellen (Session)

Dieser Befehl entspricht jenem in Kapitel 2.3.1, anstelle von Benutzername und Passwort wird SessionId und SessionKey übergeben.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"sessionkey":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 13: Signatur Request

Antwort

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

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

Listing 14: Signatur Response

2.4.4. Signatur erstellen (Session), Übergabe des Hashwertes

Dieser Befehl entspricht jenem in Kapitel 2.3.2, anstelle von Benutzername und Passwort wird SessionId und SessionKey übergeben.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 180

{
"sessionkey":"ak3k39oVApkGfZ92FhcbFmL38zK6EMunu4ooqh3foGY=",
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 15: Signatur Request (Hash)

Antwort

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

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

Listing 16: Signatur Response (Hash)

2.4.5. Signatur erstellen (Session), Übergabe des Plaintext

Dieser Befehl entspricht jenem in Kapitel 2.3.3, anstelle von Benutzername und Passwort wird SessionId und SessionKey übergeben.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/Plain HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"sessionkey":"123456789",
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 17: Signatur Request (Plain)

Antwort

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

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

Listing 18: Signatur Response (Plain)

2.4.6. Signatur erstellen (Session), JWS

Dieser Befehl entspricht jenem in Kapitel 2.3.4, anstelle von Benutzername und Passwort wird SessionId und SessionKey übergeben.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 247

{
"sessionkey":"123456789",
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 19: Signatur Request (JWS)

Antwort

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

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

Listing 20: Signatur Response (JWS)

2.5. Zertifikatsinformationen abfragen

Die Abfrage der Zertifikatsinformationen ist über einen GET Request realisiert.

Anfrage

```
GET /asignrkonline/v2/{Benutzername}/Certificate HTTP/1.1
Host: ...
```

Listing 21: Zertifikatsinformation Request

Antwort

```
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 22: Zertifikatsinformation Response

Die ersten beiden Zeilen der Antwort sind im geforderten Format für die „Beleg Gruppe“ (siehe RKSV - Anlage Detailspezifikation - Z6). Der Parameter „alg“ wird für die Payload der Signatur benötigt.

2.6. ZDA-Informationen abfragen

Die Abfrage der ZDA-Informationen ist über einen GET Request realisiert.

Anfrage

```
GET /asignrkonline/v2/{Benutzername}/ZDA HTTP/1.1
Host: ...
```

Listing 23: ZDA-Informationen Request

Antwort

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

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

Listing 24: ZDA-Informationen Response

2.7. Passwortänderung

Dieser Befehl ändert das Passwort zum Zugriff auf den a.sign RK HSM Basic/Advanced/Premium Signaturschlüssel.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Password HTTP/1.1
Content-Type: application/json
Hostr: ...
Content-Length: 53


{
  "currentpassword":"123456789",
  "newpassword":"987654321"
}
```

Listing 25: Passwortänderung Request

Antwort

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

{
  "result":true
}
```

Listing 26: Passwortänderung Response

2.8. Ausstellen eines a.sign RK HSM Basic/Advanced/Premium (Kundenkonto)

Für das Erstellen eines Kundenkontos wird ein POST Request gesendet. Dieser beinhaltet das Passwort des Partners (partner_password), die gewünschte Ausführungsvariante des Produkts und Daten für den Ordnungsbegriff. Die Antwort enthält die Zugangsdaten für das ausgestellte a.sign RK HSM Zertifikat.

Der Ordnungsbegriff (z.B.: UID-Nummer) wird in zwei Werten angegeben. Als Unterscheidung der drei möglichen Kategorien wird ein Integer angegeben, sowie der Wert selbst.

- 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

a.sign RK HSM wird in drei Ausführungsvarianten angeboten:

- product_version=1; a.sign RK HSM Basic
- product_version=2; a.sign RK HSM Advanced
- product_version=3; a.sign RK HSM Premium

Weiters ist eine e-Mail Adresse anzugeben. Diese wird zur Verständigung verwendet wenn zB das Zertifikat abläuft. Hier kann entweder die Kunden oder Partner e-Mail Adresse eingetragen werden. Es wird empfohlen keine persönliche e-Mail Adressen zu verwenden, sondern ein Postfach, welches auch bei Mitarbeiterwechsel erhalten bleibt.

Anfrage

```
POST /asignrkonline/v2/{partner_benutzername}/Account HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 261

{
"partner_password": "partnerpwd",
"classification_key_type": 0,
"classification_key": "ATU00000000",
"email": "test@test.com",
"product_version":1
}
```

Listing 27: Konto erstellen Request

Antwort

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

{
 "username":"u123456789",
 "password":"123456789",
 "Signaturzertifikat":"MIIE...QA6o=",
 "Zertifizierungsstellen":["MII...WSF"],
 "Zertifikatsseriennummer":"963244432",
 "ZertifikatsseriennummerHex":"3969F190",
 "alg":"ES256"
}
```

Listing 28: Konto erstellen Response

3. Live-System

Das Live-System zu a.sign RK HSM ist unter nachfolgenden Link erreichbar:

URL: https://www.a-trust.at/asignrkonline/v2/

Achtung, diese URL dient als Basis URL für die REST Befehle. Es müssen noch der Benutzername und der Befehl laut Kapitel 2 angehängt werden. Die Basis URL alleine führt zu einen HTTP-404 Fehler.

3.1. Zugangsdaten Partner-Funktion

Als Zugangsdaten für die Ausstellung eines a.sign RK HSM Basic/Advanced/Premium Zertifikats (siehe Kapitel 2.8) werden die Zugangsdaten für den A-Trust Webshop herangezogen. Für einen Partner mit der Partnernummer `RK 9999` kann für diese Funktion eine der folgenden URLs verwendet werden.

1. https://www.a-trust.at/asignrkonline/v2/RK%209999/Account
2. https://www.a-trust.at/asignrkonline/v2/RK9999/Account

In der URL unter 1 wird das Leerzeichen in der Partnernummer durch ein `%20` kodiert. In der URL unter 2 wurde das Leerzeichen entfernt. Beide Varianten sind funktionierende URLs für die Partnernummer `RK 9999`

3.2. Zugangsdaten für a.sign RK HSM

Die Zugangsdaten für ein Endkundenkonto des a.sign RK HSM kann entweder über die Partner-Funktionen (siehe dazu 2.8) oder direkt im Webshop erstellt werden. Für die Erstellung eines Endkundenkontos wird unter nachfolgendem Link eine Aktivierungshilfe zur Verfügung gestellt.

https://www.a-trust.at/downloads/registrierkasse/asignRKOnline_Aktivierungshilfe.pdf

4. Testsystem

Zu Testzwecken können folgende Parameter verwendet werden:

URL: https://hs-abnahme.a-trust.at/asignrkonline/v2

Benutzername: u123456789

Passwort: 123456789

Partner Benutzername: partner4711

Partner Passwort: partnerpwd

Achtung, die angegebene URL dient als Basis URL für die REST Befehle. Es müssen noch der Benutzername und der Befehl laut Kapitel 2 angehängt werden. Die Basis URL alleine führt zu einen HTTP-404 Fehler.

4.1. Weitere Testfälle

4.1.1. a.sign RK HSM - Passwort zu oft falsch eingegeben

Benutzername: u039193334

Passwort: 60z7dx

4.1.2. Partner Konto ohne Credits

Partner Benutzername: partnerNoCredits

Partner Passwort: partnerpwd

5. Implementierung des Aufrufs

5.1. Aufruf mit curl (Windows)

```
curl.exe -o outputSign.txt --cacert cas.pem -X POST -H "Content-Type:␣
    application/json" -d @requestSign.json https://.../v2/u123456789/Sign
```

Listing 29: Kommandozeilenaufruf Signatur

```
{
"password":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 30: requestSign.json

```
{
"signature":"GkXDFLK3C...feJhPwAA==",
}
```

Listing 31: outputSign.txt

```
curl.exe -o outputSign.txt --cacert cas.pem -X POST -H "Content-Type:␣
    application/json" -d @requestSign.json https://.../v2/u123456789/Sign
```

Listing 32: Kommandozeilenaufruf Zertifikatsinformationen

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

Listing 33: outputCert.txt

```
curl.exe -o outputZDA.txt --cacert cas.pem -X GET
    https://.../v2/u123456789/ZDA
```

Listing 34: Kommandozeilenaufruf Zertifikatsinformationen

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

Listing 35: outputZDA.txt

5.2. Aufruf mit C-Sharp

```
string URL = "https://.../v2/u123456789/Sign";
string request = @"{
""password"":""123456789"",
""to_be_signed"":""c2Ftc...SBzZXJ2aWNl""
}";
byte[] data = System.Text.UTF8Encoding.UTF8.GetBytes(request);


HttpWebRequest webRequest = (HttpWebRequest)WebRequest.Create(URL);
webRequest.Method = "POST";
webRequest.ContentType = "application/json";

webRequest.ContentLength = data.Length;
webRequest.GetRequestStream().Write(data, 0, data.Length);
HttpWebResponse webResponse = (HttpWebResponse)webRequest.GetResponse();

StreamReader reader = new StreamReader(webResponse.GetResponseStream(),
    System.Text.UTF8Encoding.UTF8);
string ResponseText = reader.ReadToEnd();
```

Listing 36: C# Beispiel

5.3. Aufruf mit Java

```
String urlStr = "https://.../v2/u123456789/Sign";
String request = "{\"password\":\"123456789\",␣
    \"to_be_signed\":\"c2FtcGxl...XJ2aWNl\"}";

URL url = new URL(urlStr);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
conn.setDoOutput(true);
conn.setDoInput(true);
conn.setUseCaches(false);
conn.setAllowUserInteraction(false);
conn.setRequestProperty("Content-Type", "application/json");

OutputStream out = conn.getOutputStream();
Writer writer = new OutputStreamWriter(out, "UTF-8");
writer.write(request);
writer.close();
out.close();

if (conn.getResponseCode() != 200) {
  throw new IOException(conn.getResponseMessage());
}
```

```
BufferedReader rd = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
StringBuilder sb = new StringBuilder();
String line;
while ((line = rd.readLine()) != null) {
  sb.append(line);
}
rd.close();
conn.disconnect();
String responseStr = sb.toString();
```

Listing 37: Java Beispiel

5.4. Aufruf mit PHP

```
<?php
$url = 'https://.../v2/u123456789/Sign';
$data = '{"password":"123456789","to_be_signed":"c2FtcGx...ZXJ2aWNl"}';

$options = array(
    'http' => array(
        'header'  => "Content-type: application/json\r\n",
        'method'  => 'POST',
        'content' => $data,
    ),
);

$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);

var_dump($result);
?>
```

Listing 38: PHP Beispiel

Basierend auf der Stackoverflow Antwort [?]

6. Anwendungsfälle der unterschiedlichen Signaturbefehle

Die nachfolgenden Kapitel beschreiben den Client Ablauf für die Vorbereitung der Daten, damit diese richtig an die a.sign RK HSM übergeben werden können. Die nachfolgenden Beispiele sind in Pseudocode angegeben.

6.1. Signatur

```
// JWS Header generieren
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// JWS Payload generieren
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
    ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// zu signierenden Daten generieren
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload
to_be_signed = Base64(ASCII(to_be_signed))

// a.sign RK HSM /Sign Befehl
JWS_Signature = ATrustRegMobile.Sign('user','pwd',to_be_signed)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.2. Signatur Hash

```
// JWS Header generieren
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// JWS Payload generieren
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
    ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// hash generieren
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload
hash = SHA256(ASCII(to_be_signed))
hash = Base64(hash)

// a.sign RK HSM /Sign/Hash Befehl
JWS_Signature = ATrustRegMobile.SignHash('user','pwd',hash)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.3. Signatur Plain

```
// JWS Header generieren
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// JWS Payload generieren
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0,00_0
    ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// zu signierenden Daten generieren
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload

// a.sign RK HSM /Sign/Plain Befehl
JWS_Signature = ATrustRegMobile.SignPlain('user','pwd',to_be_signed)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.4. Signatur JWS

```
// JWS Payload generieren
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0,00_0
    ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='


// a.sign RK HSM /Sign/JWS Befehl
Ergebnis = ATrustRegMobile.SignJWS('user','pwd',JWS_Payload)

// Ergebnis rueckrechnen auf Protected Header, Payload und Signatur
JWS_Protected_Header = Ergebnis.Split('.').Index(0)
JWS_Payload = Ergebnis.Split('.').Index(1)
JWS_Signature = Ergebnis.Split('.').Index(2)
```

6.5. Session Handling

```
// Session generieren
sessionid, sessionkey = ATrustRegMobile.Login('user','pwd')

// Signaturdaten vorbereiten
to_be_signed=...
Ergebnis = ATrustRegMobile.Session_Sign(sessionid, sessionkey, to_be_signed)


// Session beenden
ATrustRegMobile.Logout(sessionid)
```

A. ReturnCode der Schnittstelle

HTTP 200 Erfolgreich

HTTP 400 (invalid bas64) Base64 der Inputdaten konnte nicht konvertiert werden

HTTP 401 (invalid username or password) Benutzername oder Passwort falsch

HTTP 403 (username/session blocked) Benutzer/Session gesperrt

HTTP 401 (username/session expired) Benutzer/Session abgelaufen

HTTP 403 (username/session not supplied) Benutzername/Session nicht ausgefüllt oder zu kurz

HTTP 403 (password/sessionkey not supplied) Passwort/Sessionkey nicht ausgefüllt oder zu kurz

HTTP 500 Allgemeiner Fehler

HTTP 500 (error loading signing key) Signaturschlüssel konnte nicht geladen werden

Literatur