PoC-Signierung

Änderungshistorie

Version	Datum	Grund der Änderung, besondere Hinweise	Bearbeitung
1.0	28.10.2022	Initialversion	M. Wittig (RKI) E. Georgiew (RKI) B. Otto (RKI)

Signierung der Meldedaten

Über die Werte soll eine Signatur gebildet werden, damit die Unverfälschtheit geprüft werden kann.

Die einsendende Gesundheitseinrichtung bildet vor der Übertragung mithilfe seines auf der SmartCard befindlichen Signaturschlüssels der Telematik Infrastruktur einen signierten Hash über die Felder der Meldung.

ContentType:

Datenmodell

```
application/json
{
    "IdVersicherter": "L751666445",
    "IdDatensatz": "2020-1234",
    "IdKrankenversicherung": "108079808",
    "Signature": "0xffff",     //Signierter krytographischer Hashalgorithmus
der Daten (SHA256), die Daten werden mit dem Signaturschlüssel der TI-PKI
signiert
    "SubjectPublicKeyInfo": "0xababa"     //öffentlicher Teil des
Signaturschlüssels der einsendenden Gesundheitseinrichtung (SMB-C ORG)
}
```

Anwendungsbeispiel Hash-Bildung

```
dataToHash = "IdVersicherter:L751666445;IdDatensatz:2020-
1234;IdKrankenversicherung:108079808;Token:ey...hier_ist_der_Inhalt_des_Auth-
Tokens"
```

Bitte beachten:

- die Zeichenkette muss unter der Kodierung 'iso-8859-15' behandelt werden
- die Zeichenkette muss ohne Steuerzeichen gebildet werden (Zeilenumbrüche, ...)

HEX-Repräsentation der Zeichenkette, über die der Hash gebildet werden soll (exemplarisch)

```
00000000  49 64 56 65 72 73 69 63 68 65 72 74 65 72 3A 4C  IdVersicherter:L
00000010  37 35 31 36 36 36 34 34 35 3B 49 64 44 61 74 65  751666445;IdDate
00000020  6E 73 61 74 7A 3A 32 30 32 30 2D 31 32 33 34 3B  nsatz:2020-1234;
00000030  49 64 4B 72 61 6E 6B 65 6E 76 65 72 73 69 63 68  IdKrankenversich
00000040  65 72 75 6E 67 3A 31 30 38 30 37 39 38 30 38 3B  erung:108079808;
00000050  54 6F 6B 65 6E 3A 65 79 2E 2E 2E 68 69 65 72 5F  Token:ey...hier_
00000060  69 73 74 5F 64 65 72 5F 49 6E 68 61 6C 74 5F 64  ist_der_Inhalt_d
00000070  65 73 5F 41 75 74 68 2D 54 6F 6B 65 6E 73 00 00  es_Auth-Tokens..
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Bildung des Hash unter .NET (6)

```
byte[] dataToHash = Encoding.GetEncoding("iso-8859-
15").GetBytes("IdVersicherter:L751666445;IdDatensatz:2020-
1234;IdKrankenversicherung:108079808;Token:ey...hier_ist_der_Inhalt_des_Auth-
Tokens");
byte[] hash = SHA256.HashData(dataToHash);

var hashHexEncoded = BitConverter.ToString(hash).Replace("-", "").ToLower();
```

ECDSA-Signierzertifikat (secp256r1) für Testzwecke (DER-kodiert)

```
MIIE3QIBAzCCBJkGCSqGSIb3DQEHAaCCBIoEggSGMIIEgjCCAcsGCSqGSIb3DQEHAaCCAbwEggG4
MIIBtDCCAbAGCyqGSIb3DQEMCgECoIHMMIHJMBwGCiqGSIb3DQEMAQMwDgQI0WBSGcm/siACAgfQ
BIGoYbzQizz4n0KHfHnODQVCsBni6FT/eDQgHY2JDTc9Msz+W6faq0uP2/VK7MT4GP2u+s50Hdaa
+NolF7CWXeIN6ZunvzILzIy064FpAAfkD/AxuyWXawdP+7En0QbnCRaqdTEuBj6DOg6N1KPfVWyw
fg9ZuZlopRtsbQ0tny+TM5OjYXvOMiJoXQin+14IOvpVhiZ8IhOsz5aqqYsFCBpQ9h6oI6jyKkQM
MYHRMBMGCSqGSIb3DQEJFTEGBAQBAAAAMFsGCSqGSIb3DQEJFDFOHkwAewBEAEQAMAAzAEMAQgAx
AEYALQA0ADUAMAA3AC0ANAA3AEUANQAtADgAOABGAEUALQA0ADIAOAA0ADAAMABEADUAQwAwAEQA
QwB9MF0GCSsGAQQBgjcRATFQHk4ATQBpAGMAcgBvAHMAbwBmAHQAIABTAG8AZgB0AHcAYQByAGUA
IABLAGUAeQAgAFMAdABvAHIAYQBnAGUAIABQAHIAbwB2AGkAZABlAHIwggKvBgkqhkiG9w0BBwag
ggKgMIICnAIBADCCApUGCSqGSIb3DQEHATAcBgoqhkiG9w0BDAEDMA4ECCKWhI4UvpWjAgIH0ICC
AmhAXyjTFsAk6/Hr9SP2TVhAlWUXiXy6xpb7BkVRFgkDLxbnwm8/Y+Ukumtc91Hr3rtcEjn7HsNB
JSSzFg+TSXJ2WkPgGaJD8Kzvc2YW2BB5JmlQVzQwkjU7MLB5tsn4tgGsK6XKVkzVJ9+2ODNfBkeT
FEmRQKveaYLDHTTfSvSywUzoKeF+GqsDYNm752DE1pvjMeETCD9OkGbjQNvbEAHIqxPZONfYTHe9
vBIqRWC/khNId+eesD+ppFhDGC8QBVTOLoZW148fCjHl0e9E092nVYTCbjnrzLBN0wXjzSRjY4EF
zMFAuuuiW6mv8IUvNE9VvRkPvKm8Y2vDKHFYz1QQB7kXAyqN5ICPmoW0qM4gb929D3TElDlNi3i+
1PT45B5SJnLkUOC040FKW2whsvvoxfMmOUdyQ0yNSckXJG/zn2cRZ37q6ZUbC61p+FAdXgNyAiNh
yyY5OD2+2nmIJFPY51+LKIcjBsRRAA6MUFrfwwQYxktbSBtBggW3sK2fb/vB1IWIxbnJJ5K4whUg
Fa57FZHEMrw4njkE+38YsD+3I1BJlwvvklF4hz/lS2Fqp8EKEoQ2Rdy6yH5F2UlNI3+GOfIYsDko
NWheZWQzxXXKrjmoXCxICUFSUgtweoKPm8jQj7lVMCK1nnFIdhnVn8E8N0J+WcuNI5hFQWmWYbR8
q+Fbguaq5tZvxFfeY2eOzk47m2MykACYnnXPUTZ4fvS+lokWGCb9f+nkDtvsNrqo2ucdo4eD08Rf
+ddQcJZcb+aRHL0m5u6xaxNJ1fHBYKpOSpYh+VMy2egu+h8MGbU2wvITRpWGJZ59MDswHzAHBgUr
DgMCGgQUL2HLRGF1jvAs+x5A9G1v2fBqVUsEFK2q5llfbnqmydI/J2uVFl8DMDtPAgIH0A==
```

Signatur erzeugen (.NET 6)

```
byte[] dataToHash = Encoding.GetEncoding("iso-8859-
15").GetBytes("IdVersicherter:L751666445;IdDatensatz:2020-
1234;IdKrankenversicherung:108079808;Token:ey...hier_ist_der_Inhalt_des_Auth-
Tokens");
byte[] hash = SHA256.HashData(dataToHash);

using var certificate = new
X509Certificate2(Convert.FromBase64String(Cert_Base64_Bytes));
using var ecdsaPrivateKey = certificate.GetECDsaPrivateKey();
var signedBytes = ecdsaPrivateKey.SignHash(hash);
```

DER-Sequenz der Signatur erzeugen (.NET 6)

```
int halfLen = signedBytes.Length / 2;
var half1 = new Span<byte>(signedBytes[0..halfLen]).TrimStart((byte)0);
var half2 = new Span<byte>(signedBytes[halfLen..]).TrimStart((byte)0);

var asnWriter = new AsnWriter(AsnEncodingRules.DER);
asnWriter.PushSequence();
asnWriter.WriteIntegerUnsigned(half1);
asnWriter.WriteIntegerUnsigned(half2);
asnWriter.PopSequence();

var derSequence = asnWriter.Encode();
var derSequenceHexEncoded = BitConverter.ToString(derSequence).Replace("-",
"").ToLower();
```

Info:
Der (EC)DSA-Algorithmus ist nicht deterministisch und nutzt für die Generierung der Signatur intern einen Zufallszahlengenerator.
Somit kann hier keine eindeutige, richtige Signatur der oben genannten Werte bestimmt werden.