Anlage

Detailspezifikationen

1. Standards

Die folgenden Standards werden im Dokument unter den folgenden Abkürzungen verwendet:

- **BASE32, BASE64, BASE64-URL**: Network Working Group: Request for Comments: 4648 – The Base16, Base32, and Base64 Data Encodings
- **CRT (ICM) Mode**: NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation
- **DER**: ITU-T X.690: Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
- **JSON**: Internet Engineering Task Force (IETF): Request for Comments: 7159 – The JavaScript Object Notation (JSON) Data Interchange Format
- **JSON Web Signature**: Internet Engineering Task Force (IETF): Request for Comments: 7515 – JSON Web Signature (JWS)
- **SHA-256**: FIPS PUB 180-4 – Secure Hash Standard (SHS)
- **UTF-8**: Network Working Group: Request for Comments: 3629 – UTF-8, a transformation format of ISO 10646

2. Registrierkassenalgorithmuskennzeichen

Dieses Kennzeichen definiert die verwendeten Algorithmen und den Zertifizierungsdiensteanbieter (ZDA). Sobald ein in den Registrierkassenalgorithmuskennzeichen verwendeter Algorithmus nicht mehr im Anhang der SigV 2008 genannt wird und daher als unsicher gilt, muss ein neues Registrierkassenalgorithmuskennzeichen mit sicheren Algorithmen definiert werden und darf dieses auch bei bestehenden Registrierkassen nicht mehr eingesetzt werden. Das Kennzeichen entspricht einer Zeichenkette, die wie folgt aufgebaut ist:

RN-CM:

- „R“: Fixes Präfix
- „N“: Index für die verwendete Algorithmen-Suite startend mit 1
- „-„: Fixes Trennzeichen
- „C“: Länderkennung des ZDAs
- „M“: Index für verwendeten ZDA innerhalb der gegeben Länderkennung nach ISO 3166-1 startend mit 1

Die folgenden Kennzeichen sind definiert:

R1-CM:

- **ZDA**: CM wird als Platzhalter für die zur Verfügung stehenden ZDAs gesehen. Wenn ein geschlossenes System laut § 20 zum Einsatz kommt, muss AT0 als ZDA angebenden werden.
- **Signatur/Hashalgorithmus**: Für die Erstellung der Belegsignatur laut Z 4, Z 5 dieser Anlage. Es wird der ES256 Algorithmus nach dem JWA (JSON Web Algorithmus) Standard verwendet.
- **Hashalgorithmus für die Verkettung der Belege und Berechnung des IVs, Anzahl N der extrahierten Bytes**: Es wird SHA-256 verwendet. Die Anzahl der extrahierten und damit in den nächsten Beleg übernommen Bytes entspricht 8 (N=8).
- Kompressionsalgorithmus für kompakte Darstellung des Belegs: Dieser Algorithmus entspricht den folgenden Verfahren:
    - Aufbereitung der zu signierenden Daten: Laut Z 4, Z 5 dieser Anlage.
    - Aufbereitung des maschinenlesbaren Codes: Laut Z 12, Z 13 dieser Anlage.

3. Exportformat Datenerfassungsprotokoll

Das Exportformat des Datenerfassungsprotokolls entspricht folgender JSON-Datenstruktur:

- **Belege-Gruppe**: Der Wert dieses Feldes ist ein JSON-Array. Die Anzahl der Elemente dieses JSON-Arrays entspricht der Anzahl der Signaturzertifikate die für die Signierung der zu

exportierenden Belege verwendet wurden. Ein Element dieser Liste entspricht dabei der folgenden JSON-Datenstruktur:

- **Signaturzertifikat**: Der Wert dieses Feldes ist der BASE64-kodierte Wert des im DER-Format kodierten Signaturzertifikats.
- **Zertifizierungsstellen**: Der Wert dieses Feldes ist ein JSON-Array. Die Elemente des JSON-Arrays entsprechen der Kette aller Zertifizierungsstellen, die für die Ausstellung des Signaturzertifikats verwendet wurden. Der Wert eines Elements entspricht dem BASE64-kodierten Wert des im DER-Format kodierten Zertifikats.
- **Belege-kompakt**: Der Wert dieses Feldes ist ein JSON-Array. Die Elemente entsprechen den signierten Belegen, die in der kompakten Darstellung des JSON Web Signature Formats dargestellt werden (laut Z 6 dieser Anlage). Die Reihenfolge der Belege stimmt mit der Ablagereihenfolge im DEP überein. Es muss garantiert sein, dass die Verkettung der Signatur des Beleges an Stelle x mit dem Beleg an Stelle x+1 gegeben ist (siehe Feld „Sig-Voriger-Beleg“ laut Z 4 dieser Anlage).

4. Klartextdaten für das Signaturformat für Signierung durch Signaturerstellungseinheit

Die Signaturerstellung erfolgt nach dem JSON Web Signature (JWS) Standard. Ein Beleg ist in einer JSON-Datenstruktur abgebildet die mindestens die in § 9 Abs. 2 Z 1 bis Z 7 genannten Daten enthält. Je nach Bedarf kann das Belegformat beliebig um weitere JSON-Daten erweitert werden. Für die Transformationsverfahren die für die Signaturerstellung und die Erstellung des maschinenlesbaren Codes notwendig sind, sind aber nur die in § 9 Abs. 2 Z 1 bis Z 7 genannten Belegdaten relevant, die wie folgt in einer JSON-Datenstruktur repräsentiert werden:

- **Kassen-ID**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 1 angegebenen Wert, JSON-Format *string* UTF-8 kodiert.
- **Belegnummer**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 2 angegebenen Wert, JSON-Format *string UTF-8 kodiert*.
- **Beleg-Datum-Uhrzeit:** Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 3 angegebenen Wert, JSON-Format *string* UTF-8 kodiert. Das Datum und die Uhrzeit wird im ISO 8601 Format ohne der Angabe der Zeitzone abgespeichert („JJJJ-MM-TT’T’hh:mm:ss“, z. B. 2015-07-21T14:23:34). Es wird immer von österreichischer Lokalzeit (CET/MEZ) ausgegangen.
- **Betrag-Satz-Normal**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 4 angegebenen Wert, JSON-Format *number* mit 2 Kommastellen. Ist kein Betrag mit dieser MWST vorhanden so wird 0,00 eingetragen.
- **Betrag-Satz-Ermaessigt-1**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 4 angegebenen Wert, JSON-Format *number* mit 2 Kommastellen. Ist kein Betrag mit dieser MWST vorhanden, so wird 0,00 eingetragen.
- **Betrag-Satz-Ermaessigt-2**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 4 angegebenen Wert, JSON-Format *number* mit 2 Kommastellen. Ist kein Betrag mit dieser MWST vorhanden, so wird 0,00 eingetragen.
- **Betrag-Satz-Null**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 4 angegebenen Wert, JSON-Format *number* mit 2 Kommastellen. Ist kein Betrag ohne MWST vorhanden, so wird 0,00 eingetragen.
- **Betrag-Satz-Besonders**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 4 angegebenen Wert, JSON-Format *number* mit 2 Kommastellen. Ist kein Betrag mit dieser MWST vorhanden, so wird 0,00 eingetragen.
- **Stand-Umsatz-Zaehler-AES256-ICM**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 5 angegebenen Wert, JSON-Format *string*. BASE64-kodierter Wert des verschlüsselten Gesamtumsatzes (laut Z 8, Z 9 dieser Anlage).
- **Zertifikat-Seriennummer**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 6 angegebenen Wert, JSON-Format *string*. UTF-8 kodiert.
- **Sig-Voriger-Beleg**: Der Wert dieses Feldes entspricht dem in § 9 Abs. 2 Z 7 angegebenen Wert. JSON-Format *string*. Dieser Wert wird über die im Registrierkassenalgorithmuskennzeichen definierte kryptographische Hash-Funktion berechnet. Als Input dieser Hash-Funktion wird das Ergebnis der Signaturerstellung gemäß Z 6 verwendet. Für die Erfassung des ersten Barumsatzes wird der Wert des Felds „Kassen-ID“ als Input dieser Hash-Funktion verwendet. Aus dem Ergebnis der Hash-Funktion werden startend mit Byte 0, N Bytes extrahiert und BASE-64-kodiert. Die

Anzahl der zu extrahierenden Bytes (N) wird ebenfalls über das Registrierkassenalgorithmuskennzeichen definiert. Durch den Einsatz von Zugriffsteuerungsmethoden muss garantiert sein, dass auch bei der parallelen Abarbeitung der Belegerstellung die Verkettung über die Signaturwerte korrekt abgebildet wird.

5. Signaturformat für Signierung durch Signaturerstellungseinheit

Die zu signierenden Daten eines Belegs sind in § 9 Abs. 2 Z 1 bis Z 7 genannt. Um eine kompakte Darstellung der zu signierenden Daten zu ermöglichen, werden diese Daten in eine komprimierte Darstellung übergeführt. Die Transformation erfolgt nach der in § 9 Abs. 2 Z 1 bis Z 7 definierten Reihenfolge. Die einzelnen Felder werden UTF-8 kodiert und mit dem Zeichen „_“ zusammengeführt und in einer Zeichenkette gespeichert. Unter Verwendung der oben genannten Bezeichner und der Notation Wert(JSON-Feld) für das Extrahieren des Wertes aus der JSON-Datenstruktur des Belegs ergibt sich folgende Darstellung.

„Wert(Kassen-ID)_Wert(Belegnummer)_Wert(Beleg-Datum-Uhrzeit)_
Wert(Betrag-Satz-Normal)_Wert(Betrag-Satz-Ermaessigt-1)_Wert(Betrag-Satz-Ermaessigt-2)_Wert(Betrag-Satz-Null)_Wert(Betrag-Satz-Besonders)_
Wert(Stand-Umsatz-Zaehler-AES256-ICM)_Wert(Zertifikat-Seriennummer)_
Wert(Sig-Voriger-Beleg)“

Die resultierende Zeichenkette wird anschließend mit dem Präfix „_RKA_“ ergänzt. „RKA“ stellt einen Platzhalter für das Registrierkassenalgorithmuskennzeichen dar. Diese Kennzeichen werden in einer Liste (laut Z 2 dieser Anlage) zur Verfügung gestellt und identifizieren folgende Komponenten:

- Signatur/Hashalgorithmus für die Erstellung der Belegsignatur
- Zertifizierungsdienstanbieter (ZDA) der das Signaturzertifikat ausgestellt hat
- Hashalgorithmus für die Verkettung der Belege, sowie die Anzahl der Bytes N, die aus dem berechneten Hash-Wert extrahiert werden.
- Kompressionsalgorithmus der für die Erstellung des maschinenlesbaren Codes verwendet wurde.

Die resultierende Zeichenkette entspricht *dem Signaturformat für Signierung durch Signaturerstellungseinheit* und wird über den JSON Web Signature Standard mit dem angegeben Signaturzertifikat und dem gewählten Hashalgorithmus signiert. Im JWS-Format werden diese Daten als „JWS Payload“ bezeichnet.

6. Ergebnis der Signaturerstellung

Das Ergebnis der JWS-Signatur ist die nach dem JWS-Standard definierte kompakte Repräsentation. Diese Zeichenkette besteht dabei aus drei BASE64-URL-kodierten Elementen, die durch das Zeichen „.“ voneinander getrennt sind. Die drei Elemente entsprechen den Elementen in der gegebenen Reihenfolge

1. den Metainformationen über den verwendeten Hash bzw. Signaturalgorithmus
2. den signierten Daten (JWS Payload) und
3. dem berechneten Signaturwert.

Kann aufgrund des Ausfalls der Signaturerstellungseinheit keine digitale Signatur erstellt werden, wird statt dem berechneten Signaturwert (drittes Element der kompakten JWS Repräsentation) die UTF-8 kodierte Zeichenkette „Sicherheitseinrichtung ausgefallen“ BASE64-URL-kodiert eingetragen.

7. Anmerkung zum Wechsel des Signaturzertifikats

Bei einem Wechsel des Signaturzertifikats muss garantiert sein, dass weitere Belege nicht mehr mit dem vor dem Wechsel verwendeten Zertifikat signiert werden dürfen.

8. Verschlüsselungsmethode Umsatzzähler

Für die Verschlüsselung des Umsatzzählers wird AES-256 im ICM (CTR) Mode (Integer Counter Mode) ohne Padding verwendet. Der Initialisierungsvektor enthält einen laut Z 9 dieser Anlage berechneten Hash-Wert in dessen Berechnung die Belegnummer und die Kassenidentifikationsnummer eingeht. Der Umsatzzähler im Klartext wird in einer geeigneten Darstellung übergeben, die später auch ohne Padding-Informationen rekonstruiert werden kann.

Für die Bekanntgabe des AES-Schlüssels über FinanzOnline müssen die Binärdaten des AES-Schlüssel BASE64-kodiert werden.

9. Verschlüsselung

Für die Verschlüsselung des kodierten Umsatzzählers wird wie folgt vorgegangen:

- **Algorithmen**: Es wird der AES-256 im ICM (CTR) Mode verwendet. Für die Verschlüsselung wird kein „Padding“ verwendet.
- **Initialisierungsvektor**: Der Initialisierungsvektor (IV) für den Verschlüsselungsalgorithmus ist ein Byte-Array mit der Länge 16. Für die Berechnung des IVs werden die UTF-8 kodierte Kassenidentifikationsnummer (Wert des Feldes „Kassen-ID“ laut Z 4 dieser Anlage) und die UTF-8-kodierte Belegnummer (Wert des Feldes „Belegnummer“ laut Z 4 dieser Anlage) in dieser Reihenfolge zusammengefügt. Das Ergebnis ist eine UTF-8 kodierte Zeichenkette die als Eingabewert für die im Registrierkassenalgorithmuskennzeichen definierten Hash-Funktion verwendet wird. Das Ergebnis der Hash-Funktion ist der Hash-Wert abgebildet in einem Byte-Array. Die Bytes 0-15 werden daraus extrahiert und als IV verwendet.
  Anmerkung: Es muss garantiert sein, dass für jede Verschlüsselungsoperation, die mit einem gegebenen AES-Schlüssel durchgeführt wird, niemals der gleiche IV verwendet wird.
- **Kodierung des Umsatzwertes**: Die Block-Größe von AES-256 entspricht einem Byte-Array der Länge 16. Für die Kodierung des Umsatzzählers im Klartext wird dabei ein Byte-Array der Länge 16 erstellt. Jedes Element des Byte-Arrays wird mit 0 initialisiert. Der Umsatzzähler mit der Byte-Anzahl „N“ wird startend mit Byte 0 im BIG-ENDIAN Format als Zweier-Komplement Darstellung („signed“) gespeichert. „N“ entspricht der Anzahl der Bytes die für die Kodierung des Umsatzzählers notwendig sind. Es müssen mindestens 5 Byte lange Umsatzzähler verwendet werden.

Das Resultat der Verschlüsselung ist ein Byte-Array der Länge 16. Startend mit Byte 0 werden N Bytes aus dem Array extrahiert, BASE64-kodiert und im Beleg abgelegt.

10. Entschlüsselung

Bei der Entschlüsselung wird wie folgt vorgegangen:

- **Algorithmen**: siehe Z 8, Z 9 dieser Anlage
- **Initialisierungsvektor**: siehe Z 9 dieser Anlage
- **Aufbereitung des verschlüsselten Umsatzzählers**: Es wird ein Byte-Array der Länge 16 erstellt. Jedes Element des Byte-Arrays wird mit 0 initialisiert. Startend mit Byte 0 wird das BASE64-dekodierte Byte-Array des verschlüsselten Umsatzzählers in dem erstellen 16-Byte langem Array gespeichert.

Die aufbereiteten Daten werden mit dem AES-Algorithmus und dem AES-256 Schlüssel entschlüsselt. Das Resultat der Entschlüsselung ist ein Byte-Array der Länge 16. Startend mit Byte 0 werden N Bytes aus dem Array extrahiert und entsprechen dem entschlüsselten Umsatzzähler. Das Format entspricht dem bei der Verschlüsselung genannten „Kodierung des Umsatzwertes“.

11. Übergabeformat für Datenerfassungsprotokoll

Belege, die an das Datenerfassungsprotokoll übergeben werden, entsprechen einer JSON-Datenstruktur die mindestens folgende Werte/Daten enthalten müssen. Der Hersteller kann hier optional weitere Daten hinzufügen. Pro Beleg müssen mindestens folgende in einer JSON-Datenstruktur gespeicherten Daten verwenden werden:

- **JWS-Kompakt**: Der Wert dieses Feldes entspricht der kompakten Darstellung einer Signatur nach dem JWS-Standard (laut Z 5 dieser Anlage), JSON-Format *string*.
- **Signaturzertifikat (optional)**: Der Wert dieses Feldes ist der BASE64-kodierte Wert des im DER-Format kodierten Signaturzertifikats, JSON-Format *string*.
- **Zertifizierungsstellen (optional)**: Der Wert dieses Feldes ist ein JSON-Array. Die Elemente des JSON-Arrays entsprechen der Kette aller Zertifizierungsstellen, die für die Ausstellung des Signaturzertifikats verwendet wurden. Der Wert eines Elements entspricht dem BASE64-kodierten Wert des im DER-Format kodierten Zertifikats.

Die Werte für das Signaturzertifikat und die Zertifizierungsstellen bleiben für einen längeren Zeitraum konstant. Sie müssen daher nicht für jeden Beleg übergeben werden, sondern können auch auf einem anderen Weg dem DEP zur Verfügung gestellt werden. Es muss nur garantiert sein, dass

1. das DEP für jeden Beleg die Zuordnung zum passenden Signaturzertifikat und zu den Zertifizierungsstellen des Signaturzertifikats herstellen kann und
2. alle Zertifikate im DEP zur Verfügung stehen um den Export der signierten Belegdaten zu ermöglichen.

Für die Übergabe der Belegdaten an das DEP muss durch den Einsatz von Zugriffsteuerungsmethoden garantiert sein, dass auch bei der parallelen Abarbeitung der Belegerstellung die Verkettung über die Signaturwerte korrekt abgebildet wird (laut Z 4 dieser Anlage).

12. Details der Vorbearbeitung der im maschinenlesbaren Code enthaltenen Daten für Verifizierung des Signaturwertes eines Barumsatzes

Die für den maschinenlesbaren Code aufbereiteten Daten werden durch eine Zeichenkette repräsentiert, die folgende Elemente enthält:

- **Signierte Belegdaten**: Diese Daten entsprechen der UTF-8 kodierten Zeichenkette des Signaturformats das der Signaturerstellungseinheit übergeben wurde (laut Z 5 dieser Anlage). Die Zeichenkette kann aus dem JWS-Payload-Feld der kompakten JWS-Darstellung (Ergebnis der Signaturerstellung) extrahiert werden.
- **Signaturwert**: Der Signaturwert in BASE64-Kodierung wird aus der kompakten JWS-Darstellung (Ergebnis der Signaturerstellung) extrahiert. Es muss darauf geachtet werden, dass der Signaturwert in der kompakten Darstellung des JWS-Standards BASE64-URL-kodiert ist, um die Verwendung in Web-Anwendungen zu vereinfachen. Diese Darstellung ist aber für die QR-Code Darstellung nicht geeignet, da sie auch das Zeichen „_“ enthält, das für die Trennung der Elemente der zu signierenden Daten verwendet wird. Der BASE64-URL-kodierte Signaturwert muss daher dekodiert werden und im Standard BASE64-Format kodiert werden.

Diese zwei Elemente werden in der genannten Reihenfolge mit dem Zeichen „_“ zusammengesetzt, UTF-8 kodiert und in einem maschinenlesbaren Code aufbereitet.

13. Prüfung des maschinenlesbaren Codes

Die Prüfung der Signatur, die in einem maschinenlesbaren Code aufbewahrt wird, wird wie folgt durchgeführt:

1. **Lesen des maschinenlesbaren Codes**: Die gelesene UTF8-kodierte Zeichenkette enthält die „Signierten Belegdaten“ und den „Signaturwert“.
2. **Extraktion der „Signierten Belegdaten“ und des „Signaturwerts“**: Die „Signierten Belegdaten“ und der „Signaturwert“ werden aus der UTF8-kodierten Zeichenkette über das Trennzeichen „_“ extrahiert. Der BASE64-kodierte Signaturwert wird BASE64-dekodiert.
3. **Aufbereitung der kompakten Darstellung anhand des JWS-Signatur-Standards**: Die kompakte Darstellung (laut Z 5 dieser Anlage) wird wie folgt aus dem maschinenlesbaren Code rekonstruiert. Die einzelnen Elemente werden dabei durch das Zeichen „.“ zusammengeführt.
   a) **JWS Protected Header:** Der Signatur/Hashalgorithmus des JWS Protected Headers kann über das Registrierkassenalgorithmuskennzeichen rekonstruiert werden. Der JWS Protected Header wird UTF-8-kodiert in der Zeichenkette an der 1. Stelle BASE64-URL-kodiert abgespeichert.
   b) **JWS Payload**: Die JWS Payload entspricht den zuvor extrahierten Belegdaten und wird in der Zeichenkette an der 2. Stelle BASE64-URL-kodiert abgespeichert.
   c) **JWS Signature**: Dieser Wert entspricht dem vorher extrahierten Signaturwert und wird in der Zeichenkette an der 3. Stelle BASE64-URL-kodiert abgespeichert.
4. **Prüfen der Signatur**: Die aufbereitete kompakte Darstellung anhand des JWS-Standards wird mit dem entsprechenden Signaturzertifikat geprüft.

14. Erstellung der OCR-fähigen Zeichenkette

Für die OCR-fähige Zeichenkette wird aufgrund der Schwierigkeit, alle möglichen Zeichen einer BASE64 Zeichenfolge automatisiert und bei realistischen Lichtbedingungen und Kameraeigenschaften sicher zu erkennen, statt der BASE64-Darstellung der folgenden Elemente laut Z 4, Z 12 dieser Anlage die BASE32-Darstellung der Binärdaten gewählt.

- Signaturwert
- Sig-Voriger-Beleg
- Stand-Umsatz-Zaehler-AES256-ICM

Die resultierende Zeichenkette wird im OCR-A Font auf den Beleg gedruckt.

15. Prüfung der OCR-fähigen Zeichenkette

Für die Prüfung der OCR-fähigen Zeichenkette müssen die BASE32-kodierten Elemente auf die BASE64-Kodierung umkodiert werden. Der anschließende Prüfvorgang ist äquivalent zum Prüfen des maschinenlesbaren Codes.

16. OID

Der OID-Bezeichner für die Verwendung im Signaturzertifikat entspricht 1.2.40.0.10.1.11.1 „Österreichische Finanzverwaltung Registrierkasseninhaber“.

Die OID wird aus dem Teilbaum 1.2.40.0.10.1.11 „Teilbaum Bundesministerium für Finanzen“ vergeben.