Bundesverband der Deutschen Volksbanken und Raiffeisenbanken e. V.
Bundesverband deutscher Banken e. V.
Bundesverband Öffentlicher Banken Deutschlands e. V.
Deutscher Sparkassen- und Giroverband e. V.
Verband deutscher Pfandbriefbanken e. V.

Die Deutsche Kreditwirtschaft

FinTS

Financial Transaction Services

Schnittstellenspezifikation

Security

Sicherheitsverfahren HBCI (inklusive Secoder)

Herausgeber:
Bundesverband deutscher Banken e.V., Berlin
Deutscher Sparkassen- und Giroverband e.V., Bonn/Berlin
Bundesverband der Deutschen Volksbanken und Raiffeisenbanken e.V., Berlin
Bundesverband Öffentlicher Banken Deutschlands e.V., Berlin

Version: 4.1 FV Stand: 29.11.2018 Final Version

Die vorliegende Schnittstellenspezifikation für eine automatisiert nutzbare multibankfähige Banking-Schnittstelle (im Folgenden: Schnittstellenspezifikation) wurde im Auftrag der Deutschen Kreditwirtschaft entwickelt. Sie wird hiermit zur Implementation in Kunden- und Kreditinstitutssysteme freigegeben.

Versionsführung

Das vorliegende Dokument wurde von folgenden Personen erstellt bzw. geändert:

Name	Organisation	Datum	Version	Dokumente	Anmerkungen
Haubner	für SIZ	04.12.2012	4.1 draft 01	FinTS_4.1_Security_HBCI_D01.doc	Anpassungen für FinTS V4.1
Haubner	für SIZ	20.01.2014	4.1 Final Version	FinTS_4.1_Security_HBCI_2014-01-20_FV.doc	Klarstellungen und Fehlerkorrekturen
Haubner	für SIZ	29.11.2018	4.1 Final Version	FinTS_4.1_Security_HBCI_2018-11-29_FV.doc	Anpassungen wegen PSD2 und RTS

Änderungen gegenüber der Vorversion

Änderungen sind im Dokument durch einen Randbalken markiert. Hypertextlinks sind in dieser Farbe markiert. Falls sich die Kapitelnummerierung geändert hat, bezieht sich die Kapitelangabe auf die neue Nummerierung.

lfd. Nr.	Kapitel	Kapitelnummer	Kennung[1]	Art[2]	Beschreibung
1	Abkürzungen			E	Einfügen neuer Abkürzungen
2	Literaturhinweise			E	Einfügen und Aktualisieren von Literaturhinweisen
3	Einleitung, Verfahrensbeschreibung	I und II		E	Einführen des RAH-Verfahrens und der damit verbundenen Sicherheitsprofile RAH-7, RAH-9 und RAH-10 Entfernen der RDH-Verfahren und des DDV-Verfahrens
4	Secoderintegration	III		E	Ergänzen eines Kapitels zur Secoderintegration
5	Chipapplikationen	IV		Ä	Ergänzen / Ersetzen der Chipapplikationen für SECCOS
6	II.1.1, S. 2			Ä	Änderung des Passus zu verpflichtenden Sicherheitsprofilen.
7	II.2.1, S.11			Ä	ZKA-Padding, einfügen der AES-Blocklänge=16 Byte für den Wert „L“ Fehlerbehebungen und Klarstellungen in den Abbildungen 1, 2 und 3
8	II.3.1.2.1, S.17			Ä	Löschen von Step 5, da nicht mehr relevant.
9	Diverse			Ä	Ersetzen der konkret angegebenen Schlüssellängen durch Referenz auf die Empfehlungen des DK Kryptokatalogs [DK Krypto]

[1] nur zur internen Zuordnung
[2] F = Fehler; Ä = Änderung; K = Klarstellung; E = Erweiterung

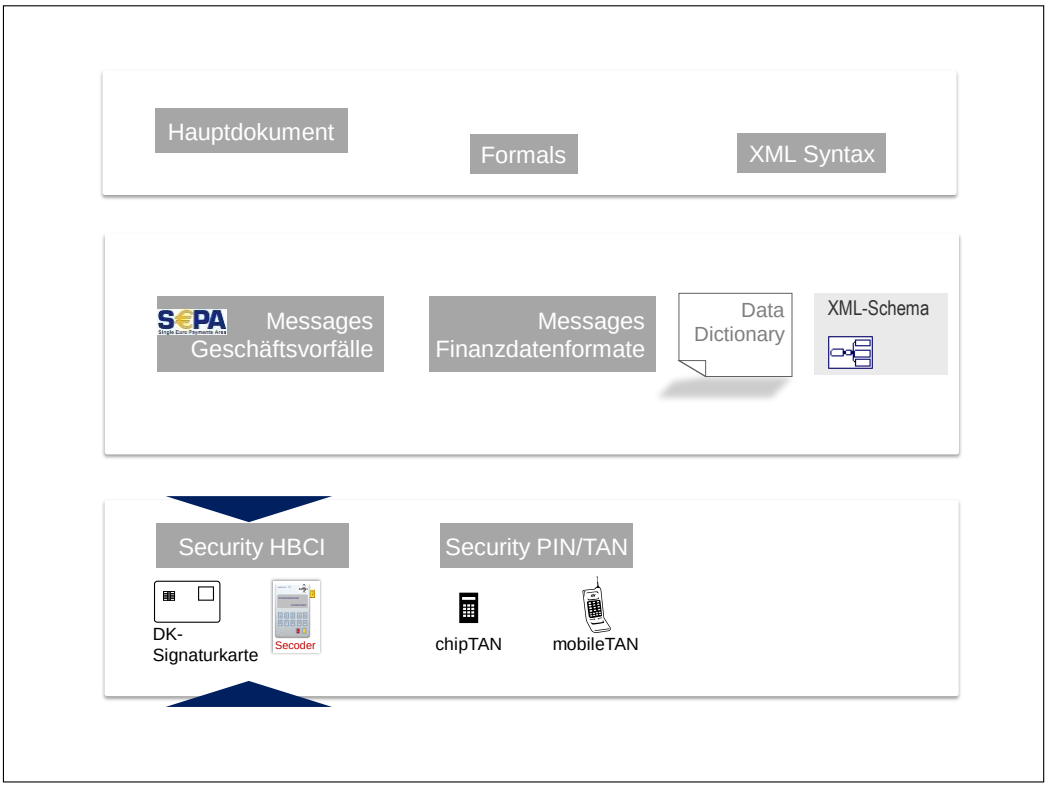
lfd. Nr.	Kapitel	Kapitel-nummer	Kennung[1]	Art[2]	Beschreibung
10	Diverse			Ä	Entfernen aller Bezüge zu SECCOS-5
11	Terminalabläufe			Ä	Anpassen für RAH-Algorithmen, Entfernen V001 für RIPEMD-160 in EF_NOTEPAD

Releasedatum 29.11.2018

lfd. Nr.	Kapitel	Kapitel-nummer	Kennung	Art	Beschreibung
1	Verfahrensbeschreibung	B	0490	Ä/K	Berücksichtigung von PSD2-Anforderungen bei einer Neustrukturierung der unterstützten Sicherheits-Verfahren und –Mechanismen. Entfernen von nicht mehr unterstützten Sicherheitsprofilen.
2	Verfahrensbeschreibung	B	0490	Ä	Festlegen der maximalen RSA-Schlüssellängen auf bis zu 2048 Bit.
3	Chipapplikationen	C	0490	Ä	Löschen der Chipkartenapplikation „DF_BANKING“
4	Chipapplikationen	C.1.2.1	0490	Ä	Löschen der EF-NOTPAD Version 001.

Dokumentenstruktur

Das vorliegende Dokument steht in folgendem Bezug zu den anderen Bänden der FinTS-Spezifikation:



Dokumenteninhalte, Abkürzungen, Definitionen und Literaturhinweise befinden sich im FinTS Hauptdokument [Master].

Inhaltsverzeichnis

Abbildungsverzeichnis

I. EINLEITUNG

In diesem Dokument wird das Sicherheitsverfahren HBCI beschrieben Dieses Verfahren beruht auf modernen kryptographischen Methoden und Algorithmen, wie sie auch durch [PSD2] gefordert sind (z. B. der Digitalen Signatur mit Kryptoverfahren nach Stand der Technik und Chipkartentechnologie).

In Abschnitt III ist die Integration des Secoders beschrieben, wie sie ab FinTS V4.1 als neue Funktionalität in das Sicherheitsverfahren HBCI integriert wurde.

Dieses Sicherheitsverfahren kann in multibankfähigen Online-Banking-Verfahren der Deutschen Kreditwirtschaft eingesetzt werden.

Diese Spezifikation enthält die Formate für Signatur, Verschlüsselung und Keymanagement. Informationen bzgl. Nachrichtenaufbau und Kommunikationsablauf sind dem Dokument [Formals] zu entnehmen.

II. VERFAHRENSBESCHREIBUNG

II.1 Allgemeines

Die Verfahrensbeschreibung ist in folgende sechs Abschnitte unterteilt:

1. Allgemeines, Sicherheitsprofile und Sicherheitsklassen
2. Verwendete Sicherheitsmechanismen
3. Abläufe
4. Bankfachliche Anforderungen
5. Segmentformate für Signatur und Verschlüsselung
6. Key-Management

Grundsätzlich kommen im Rahmen von HBCI Sicherheitslösungen zum Einsatz, die auf dem asymmetrischen RSA-Verfahren basieren.

Dabei existieren zwei Varianten, die mit RAH (RSA-AES-Hybridverfahren) und einer Sicherheitsprofilnummer gekennzeichnet werden. RAH verwendet RSA-Signaturen und chiffriert den Nachrichtenschlüssel mittels RSA. Die Daten-Verschlüsselung wird durch eine AES-Verschlüsselung mit dem Nachrichtenschlüssel erreicht. Als Träger der privaten Schlüssel dienen Bankensignaturkarten auf Basis des SECCOS-Betriebssystems der Deutschen Kreditwirtschaft oder Softwareschlüssel, die mit einem Kennwort gesichert sind.

II.1.1 Dynamic Linking und Transparenz der zu signierenden Daten

Die Verwendung von RSA-Signaturen sorgt bei HBCI für die Bildung von Authentifizierungscodes gemäß den Anforderungen aus [PSD2] und [RTS]. Die Forderung nach Transparenz relevanter Daten wie z. B. Empfänger-IBAN und Betrag bei Zahlungsverkehrstransaktionen bzgl. Integrität, Authentizität und Vertraulichkeit kann bei HBCI auf zwei Arten erreicht werden:

1. Verwenden eines Secoders im Applikationsmodus „aut“.
2. Einsatz einer Software-Komponente für einen entsprechenden Schutz bei der Anzeige der zu signierenden Daten. Hierbei kann es sich auch um eine separate Anzeige (z. B. eigenes Fenster) in einem FinTS-Kundenprodukt handeln.

Die Verwendung des Secoder-Applikationsmodus in Kombination mit HBCI ist in [Secoder] beschrieben. Für beide Anwendungsszenarien werden im Folgenden geeignete Sicherheitsprofile beschrieben.

II.1.2 Sicherheitsprofile

Die RAH-Sicherheitsverfahren können unterschiedlich parametrisiert werden, wobei Sicherheitsprofile entstehen. Um Multibankfähigkeit zu gewährleisten, ist bei Kommunikation auf Basis von FinTS 4.1 kundenproduktseitig die Unterstützung der Sicherheitsprofile RAH-7 und RAH-9 verpflichtend. Zusätzlich kann auch das Sicherheitsprofil RAH-10 für SW-basierte Schlüssel verwendet werden. Andere als die unten genannten Sicherheitsprofile sind nicht zulässig.

Das Kreditinstitut teilt dem Kunden die bankseitig unterstützten Profile in den Bankparameterdaten mit. Der Kunde wählt aus diesen Verfahren das für ihn geeignete

Verfahren aus und bildet auf diese Weise Signatur und Verschlüsselung. Das Kreditinstitut antwortet stets mit dem vom Kunden gewählten Verfahren.

Hier eine Übersicht der zugelassenen Sicherheitsprofile und deren Anwendungsspektrum:

Sicherheits-profil	Secoder Option	Schlüssel-länge	Medium	Bemerkungen
RAH-7	811	..2048	Bankensignaturkarte ≥ SECCOS 6	mit SHA-256, PKCS#1 PSS Padding, AES-Verschlüsselung
RAH-9	811	..2048	Bankensignaturkarte ≥ SECCOS 6	wie RAH-7 ohne Zertifikate
RAH-7		..2048	Bankensignaturkarte ≥ SECCOS 6	mit SHA-256, PKCS#1 PSS Padding, AES-Verschlüsselung
RAH-9		..2048	Bankensignaturkarte ≥ SECCOS 6	wie RAH-7 ohne Zertifikate
RAH-10		..2048	gehärtete SW-Komponente zur Schlüsselablage	mit SHA-256, PKCS#1 PSS Padding, AES-Verschlüsselung

Bei Verwendung der Secodervisualisierung hat die *SecoderOption* gemäß [Secoder] folgende Bedeutung:

811 Signatur gemäß HBCI-Sicherheitsprofil RAH-7 bzw. RAH-9 unter Verwendung einer Bankensignaturkarte und der Secoderanwendung *aut*. Die Visualisierung erfolgt im Secoderdisplay analog den Angaben in der BPD-Parameterstruktur *SecoderSignatureParam.*

Die Angaben zum SECCOS-Betriebssystem bzw. der Betriebssystemversion sind nur als beispielhaft anzusehen; es kann auch jede gleichwertige Signaturkarte verwendet werden, welche die geforderten Verfahren unterstützt.

Die Information über die Betriebssystemversion kann dem Byte 24 in EF_ID entnommen werden. Dort ist derzeit folgender Wert vorgesehen:

X’06’: SECCOS 6

X’07’: SECCOS 7

II.1.3 Kartenbasierte Sicherheitsprofile im Secoder-Applikationsmodus

Für den Einsatz der folgenden Sicherheitsprofile ist als Chipkartenleser ein Secoder mindestens in Version 2.1 Voraussetzung.

- **RAH-7 im Secoder-Applikationsmodus (SecoderOption=811)**

  Als Sicherheitsmedium für das Kundensystem ist nur die Bankensignaturkarte oder eine gleichwertige Signaturkarte zugelassen. Als Chipkartenleser ist ein Secoder ab Version 2.1 zu verwenden. Im Applikationsmodus des Secoders haben Signaturen z. B. bei Sicherheitsfunktion 811 folgenden Aufbau:

Parameter	Bedeutung/ Anmerkung	Wert (URI)
Signaturalgorithmus	mit Signier-schlüssel: RSA, Padding nach RSASSA-PSS, SHA-256/SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2
	mit Signatur-schlüssel: RSA, Padding nach RSASSA-PSS, SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256
Hashalgorithmus	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256
Verschlüsselungsal-gorithmus	AES-256	http://www.fints.org/spec/xmlschema/4.1/xmlenc#aes256
Schlüsselart	Signierschlüssel Chiffrierschlüssel Schlüssel für Digi-tale Signaturen	S, C, D
Verschlüsselungsal-gorithmus Einmal-schlüssel	RSA (inkl. PKCS#1-Padding)	http://www.w3c.org/2001/04/xmlenc#rsa-1_5
Schlüssellänge		
Zertifikatstyp	X.509	
Zertifikatsinhalt	EF_X509.CH.DS	fortgeschritten, gemäß Sicherheitsklasse

Im Rahmen des Paddingverfahrens RSASSA-PSS wird als „Mask Generation Function“ MGF1 verwendet. Beim Signierschlüssel wird ein doppeltes Hashing (Software und Bankensignaturkarte) durchgeführt. Dies wird gekennzeichnet durch den speziellen Signaturalgorithmus
`http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2`.

Als Salt-Länge (Länge des Initialwertes) ist die Länge des Hashwertes zu verwenden. Diese Festlegung ist z. B. auch Bestandteil der SECCOS Spezifikation [SECCOS-6] bzw. [SECCOS-7].

- **RAH-9 im Secoder-Applikationsmodus (SecoderOption=811)**

  Als Sicherheitsmedium für das Kundensystem ist nur die Bankensignaturkarte oder eine gleichwertige Signaturkarte zugelassen. Als Chipkartenleser ist ein Secoder ab Version 2.1 zu verwenden. Im Applikationsmodus des Secoders haben Signaturen z. B. bei Sicherheitsfunktion 811 folgenden Aufbau:

Parameter	Bedeutung/ Anmerkung	Wert (URI)
Signaturalgorithmus	mit Signier-schlüssel: RSA, Padding nach RSASSA-PSS, SHA-256/SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2
Hashalgorithmus	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256
Verschlüsselungsal-gorithmus	AES-256	http://www.fints.org/spec/xmlschema/4.1/xmlenc#aes256
Schlüsselart	Signierschlüssel Chiffrierschlüssel	S, C
Verschlüsselungsal-gorithmus Einmal-schlüssel	RSA (inkl. PKCS#1-Padding)	http://www.w3c.org/2001/04/xmlenc#rsa-1_5
Schlüssellänge		
Zertifikatstyp		ohne
Zertifikatsinhalt	nicht spezifiziert	

Bei Einsatz von RSASSA-PSS wird beim Signierschlüssel ein doppeltes Hashing (Software und Bankensignaturkarte) durchgeführt. Dies wird gekennzeichnet durch den speziellen Signaturalgorithmus `http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2`.

Als Salt-Länge (Länge des Initialwertes) ist die Länge des Hashwertes zu verwenden. Diese Festlegung ist z. B. auch Bestandteil der SECCOS Spezifikation [SECCOS-6] bzw. [SECCOS-7].

II.1.4 Kartenbasierte Sicherheitsprofile ohne Secoder-Applikationsmodus

- **RAH-7**

  Als Sicherheitsmedium für das Kundensystem ist nur die Bankensignaturkarte oder eine gleichwertige Signaturkarte zugelassen.

Parameter	Bedeutung/ Anmerkung	Wert (URI)
Signaturalgorithmus	mit Signierschlüssel: RSA, Padding nach RSASSA-PSS, SHA-256/SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2
	mit Signaturschlüssel: RSA, Padding nach RSASSA-PSS, SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256
Hashalgorithmus	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256
Verschlüsselungsalgorithmus	AES-256	http://www.fints.org/spec/xmlschema/4.1/xmlenc#aes256
Schlüsselart	Signierschlüssel Chiffrierschlüssel Schlüssel für Digitale Signaturen	S, C, D
Verschlüsselungsalgorithmus Einmalschlüssel	RSA (inkl. PKCS#1-Padding)	http://www.w3c.org/2001/04/xmlenc#rsa-1_5
Schlüssellänge		
Zertifikatstyp	X.509	
Zertifikatsinhalt	EF_X509.CH.DS	fortgeschritten, gemäß Sicherheitsklasse

Bei Einsatz von RSASSA-PSS wird beim Signierschlüssel ein doppeltes Hashing (Software und Bankensignaturkarte) durchgeführt. Dies wird gekennzeichnet durch den speziellen Signaturalgorithmus `http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2`.

Als Salt-Länge (Länge des Initialwertes) ist die Länge des Hashwertes zu verwenden. Diese Festlegung ist z. B. auch Bestandteil der SECCOS Spezifikation [SECCOS-6] bzw. [SECCOS-7].

- **RAH-9**

  Als Sicherheitsmedium für das Kundensystem ist nur die Bankensignaturkarte oder eine gleichwertige Signaturkarte zugelassen.

Parameter	Bedeutung/ Anmerkung	Wert (URI)
Signaturalgorithmus	mit Signierschlüssel: RSA, Padding nach RSASSA-PSS, SHA-256/SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2
Hashalgorithmus	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256
Verschlüsselungsalgorithmus	AES-256	http://www.fints.org/spec/xmlschema/4.1/xmlenc#aes256
Schlüsselart	Signierschlüssel Chiffrierschlüssel	S, C
Verschlüsselungsalgorithmus Einmalschlüssel	RSA (inkl. PKCS#1-Padding)	http://www.w3c.org/2001/04/xmlenc#rsa-1_5
Schlüssellänge		
Zertifikatstyp		ohne
Zertifikatsinhalt	nicht spezifiziert	

Bei Einsatz von RSASSA-PSS wird beim Signierschlüssel ein doppeltes Hashing (Software und Bankensignaturkarte) durchgeführt. Dies wird gekennzeichnet durch den speziellen Signaturalgorithmus `http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2`.

Als Salt-Länge (Länge des Initialwertes) ist die Länge des Hashwertes zu verwenden. Diese Festlegung ist z. B. auch Bestandteil der SECCOS Spezifikation [SECCOS-6] bzw. [SECCOS-7].

II.1.5 SW-basiertes Sicherheitsprofil

- **RAH-10**

  Als Sicherheitsmedium für das Kundensystem ist eine RSA-Softwarelösung unter folgenden Rahmenbedingungen zugelassen.

Parameter	Bedeutung/ Anmerkung	Wert (URI)
Signaturalgorithmus	mit Signierschlüssel: RSA, Padding nach RSASSA-PSS, SHA-256/SHA-256	http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2
Hashalgorithmus	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256
Verschlüsselungsalgorithmus	AES-256	http://www.fints.org/spec/xmlschema/4.1/xmlenc#aes256
Schlüsselart	Signierschlüssel Chiffrierschlüssel	S, C
Verschlüsselungsalgorithmus Einmalschlüssel	RSA (CBC-0-Padding)	http://www.w3c.org/2001/04/xmlenc#rsa
Schlüssellänge		
Zertifikatstyp	X.509	
Zertifikatsinhalt	nicht spezifiziert	

Im Rahmen des Paddingverfahrens RSASSA-PSS wird als „Mask Generation Function“ MGF1 verwendet. Beim Signierschlüssel wird ein doppeltes Hashing (Software und Bankensignaturkarte) durchgeführt.
Dies wird gekennzeichnet durch den speziellen Signaturalgorithmus `http://www.fints.org/spec/xmlschema/4.1/xmldsig#rsa-rsassapss-sha256x2`.

Als Salt-Länge (Länge des Initialwertes) ist die Länge des Hashwertes zu verwenden.

II.1.6 Sicherheitsklassen

Die Sicherheitsklasse gibt für jede Signatur den erforderlichen Sicherheitsdienst an. Als Sicherheitsdienst gelten derzeit „Authentifikation“ und „Non-Repudiation“.

Der Sicherheitsdienst „Authentifikation“ erfordert die Signatur mit dem Signierschlüssel (Schlüsselart „S“; Schlüssel auf Benutzerseite: SK.CH.AUT/KE). Der Sicherheitsdienst „Non-Repudiation“ erfordert die Signatur mit dem Schlüssel für Digitale Signatur (Schlüsselart „D“; Schlüssel auf Benutzerseite: SK.CH.DS).

Derzeit sind folgende Sicherheitsklassen zulässig:

Code	Bedeutung
0	kein Sicherheitsdienst erforderlich
1	Sicherheitsdienst „Authentikation“
2	Sicherheitsdienst „Authentikation“ mit fortgeschrittener elektronischer Signatur gemäß §2, SigG und optionaler Zertifikatsprüfung unter Verwendung des S-Schlüssels (Schlüssel S_K.CH.$AUT_{C/S}$)
3	Sicherheitsdienst „Non-Repudiation“ mit fortgeschrittener elektronischer Signatur gemäß §2, SigG und optionaler Zertifikatsprüfung unter Verwendung des DS-Schlüssels (S_K.CH.DS)
4	Sicherheitsdienst „Non-Repudiation“ mit fortgeschrittener bzw. qualifizierter elektronischer Signatur gemäß §2, SigG und zwingender Zertifikatsprüfung unter Verwendung des DS-Schlüssels (S_K.CH.DS)

Folgende Zuordnungen von Sicherheitsklassen auf Sicherheitsprofile sind möglich:

Sicherheitsprofil	Sicherheitsklasse(n)
RAH-7	1, 2, 3, 4
RAH-9	1, 2
RAH-10	1

Die Sicherheitsklasse gibt für jeden Geschäftsvorfall den erforderlichen Sicherheitsdienst an. Signaturen gemäß der Sicherheitsklasse 2 und höher entsprechen den Anforderungen des Signaturgesetzes und erlauben damit rechtsverbindliche Willenserklärungen unter der Voraussetzung, dass die außerhalb des FinTS-Protokolls liegenden Anforderungen (z. B. Anforderungen an die Zertifizierungsinfrastruktur und an die Endgeräte) ebenfalls erfüllt sind.

Jede Signatur, die im Rahmen von HBCI-Sicherheit generiert wird, muss mindestens der festgelegten Sicherheitsklasse entsprechen:

- Technische Signaturen (Botensignaturen) können generell mit Sicherheitsklasse 1 (Authentifikation) erfolgen.
- Für Herausgebersignaturen von Geschäftsvorfällen kann das Kreditinstitut die mindestens notwendige Sicherheitsklasse individuell festlegen (Die Sicherheitsklasse wird dem Benutzer in den Bankparameterdaten des betreffenden Geschäftsvorfalls mitgeteilt)

Hinweis:

Sicherheitsklassen werden nur in Verbindung mit dem Sicherheitsverfahren HBCI benutzt. Unterstützt ein Kreditinstitut ausschließlich das PIN/TAN-Verfahren, so ist in das DE ‚Sicherheitsklasse' des jeweiligen Geschäftsvorfallparametersegmentes als Füllwert ‚0' einzustellen. Die Sicherheitsklasse hat bei PIN/TAN für die Verarbeitung keine Bedeutung und darf vom Kundenprodukt für PIN/TAN nicht ausgewertet werden. Stattdessen sind bei PIN/TAN die Informationen aus HIPINS für die Festlegung benötigter Sicherheitsmerkmale zu verwenden.

II.2 Mechanismen

Dieser Abschnitt beschreibt die algorithmischen Grundlagen der Sicherheitsmechanismen. Die Einbindung der hier beschriebenen Vorgänge in das FinTS-Protokoll ist in *II.5 Formate für Signatur und Verschlüsselung* und in [Syntax] beschrieben.

II.2.1 Elektronische Signatur

Die Bildung der elektronischen Signatur erfolgt durch die Vorgänge

- Bildung des Hash-Wertes
- Ergänzen des Hash-Wertes auf eine vorgegebene Länge und
- Berechnung der elektronischen Signatur über den Hash-Wert.

Je nach Sicherheitsverfahren sind die Verarbeitungsschritte jeweils verschieden.

II.2.1.1 Hashing

Als Hash-Funktion kommt im Rahmen von HBCI-Sicherheit derzeit ausschließlich SHA-256 [SHA-256] zum Einsatz.

♦ **SHA-256**

Der Hash-Algorithmus SHA-256 bildet Eingabe-Bitfolgen beliebiger Länge auf Bytefolgen von 32 Byte Länge ab. Teil des Hash-Algorithmus ist das Padding von Eingabe-Bitfolgen auf ein Vielfaches von 64 Byte. Das Padding erfolgt auch dann, wenn die Eingabe-Bitfolge bereits eine Länge hat, die ein Vielfaches von 64 Byte ist. SHA-256 verarbeitet die Eingabe-Bitfolgen in Blöcken von 64 Byte Länge.

II.2.1.2 Elektronische Signatur bei RAH (RSA-basierend)

1. Hashing der Nachricht

Als Hash-Funktion wird SHA-256 eingesetzt.

2. Formatierung des Hash-Wertes

Die Formatierung des Hash-Wertes erfolgt gemäß PKCS#1 PSS.

3. Berechnung der elektronischen Signatur

Der Hash-Wert wird mittels RSA gemäß PKCS#1 PSS signiert.

II.2.2 Verschlüsselung

Bei jeder Verschlüsselung wird ein separater Einmalschlüssel verwendet. Die Verschlüsselung der FinTS-Nutzdaten erfolgt mittels AES-256 gemäß [AES]. Der Nachrichtenschlüssel wird mittels RSA (RAH) chiffriert und mit der verschlüsselten Nachricht mitgeliefert.

Der Einmalschlüssel (=Nachrichtenschlüssel) muss für jede Verschlüsselung innerhalb einer FinTS-Kommunikation individuell verschieden sein. Dies muss gewährleistet werden, indem das sendende System den Einmalschlüssel dynamisch generiert.

Sollte bei der Verarbeitung des Nachrichtenschlüssels, insbesondere beim Padding ein Fehler auftreten, so sind außer dem negativen Prüfergebnis selbst keine weiteren Details an die aufrufende Funktion zurückzugeben, um keine Rückschlüsse über die Art des Fehlers und damit ggf. auf den Schlüssel selbst zu geben.

II.2.2.1 Verschlüsselung bei RAH-7, RAH-9 und RAH-10:

Die Verschlüsselung und Entschlüsselung erfolgt bei den RAH-Verfahren in den folgenden drei Schritten:

1. Der Sender erzeugt eine Zufallszahl als Einmalschlüssel.
2. Dieser Einmalschlüssel wird verwendet, um die FinTS-Nutzdaten mittels AES im CBC Modus gemäß ISO 10116 (ANSI X3.106) [ISO 10116]/[X3.106] zu verschlüsseln (vgl. Abbildung 1). Das Padding der Nachricht erfolgt gemäß den Vorgaben des Kryptokatalogs der Deutschen Kreditwirtschaft (vgl. [DK Krypto], Kapitel 4.3.1) (vgl. *Abbildung 2* und *Abbildung 3*).

„ZKA-Padding“ (vgl. [DK Krypto], Kapitel 4.3.1 auf S. 20):

Für die Verarbeitung von Daten durch einen kryptographischen Algorithmus kann deren Darstellung als Folge von Byte-Blöcken mit einer vorgegebenen Länge L erforderlich sein. Das ZKA-Padding ist eine Methode zur Formatierung des letzten, möglicherweise unvollständigen Datenblocks auf die Länge von L Byte. Die den Daten zugehörigen Bytes können eindeutig von den durch das Padding hinzugefügten Bytes unterschieden werden.

An die Daten M wird zunächst das Byte '80' angehängt. Falls M || '80' nun eine Byte-Länge besitzt, die kein Vielfaches von L ist, werden weitere Bytes '00' angehängt, bis das Ergebnis der Operation eine Byte-Länge besitzt, die ein Vielfaches von L ist.

$$\text{ZKA-Padding}(M) = M \,||\, '80' \,||\, \underbrace{'00' \,||\ldots||\, '00'}_{\text{Verkettung bis zur Gesamtlänge der Byte-Folge als Vielfaches von L Byte (hier: AES-Blocklänge = 16 Byte)}}$$

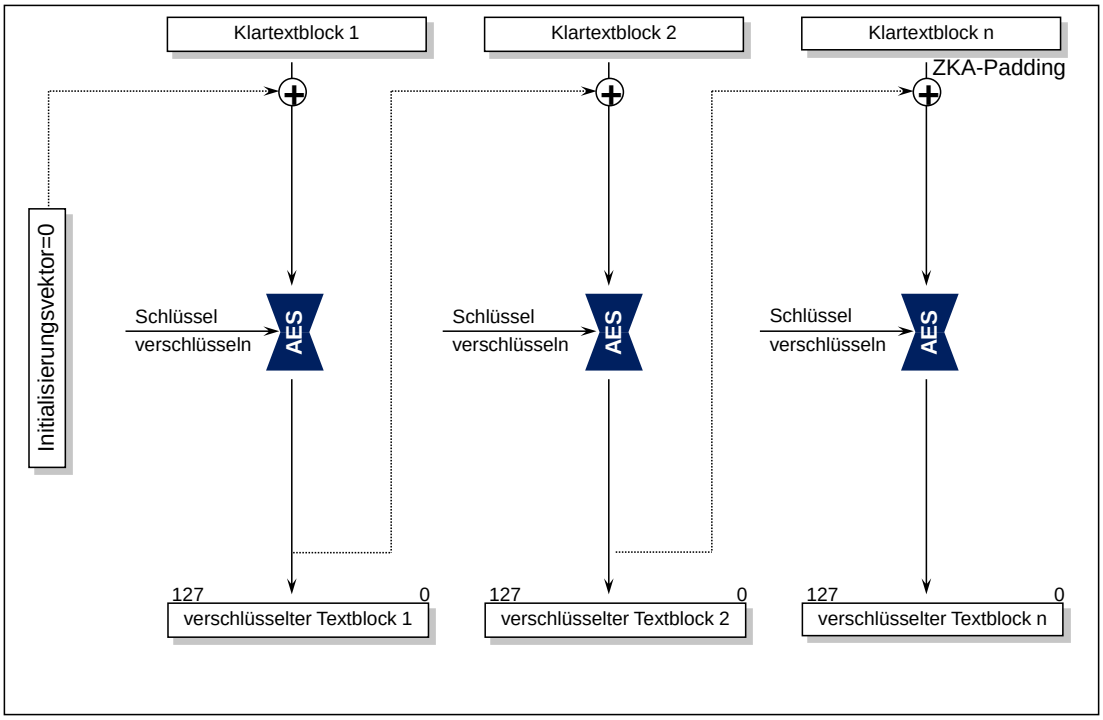


Abbildung 1: Nachrichtenverschlüsselung mit AES im CBC-Mode für RAH-Verfahren

Hinweis zu den Abbildungen: Die Angabe der Längen erfolgt in Form von Bitpositionen (z. B. 127 … 0), um grafisch zu zeigen, an welcher Stelle das Padding erfolgt.

3. Der aktuelle Einmalschlüssel wird mit dem öffentlichen Schlüssel des Empfängers chiffriert. Da die Länge des Einmalschlüssels bei AES nur 32 Byte, d.h. 256 Bit beträgt, muss er auf die Moduluslänge des verwendeten öffentlichen Chiffrierschlüssels ergänzt werden. Das Padding wird abhängig vom Sicherheitsprofil auf unterschiedliche Art und Weise vorgenommen, wie in den folgenden Abbildungen gezeigt.

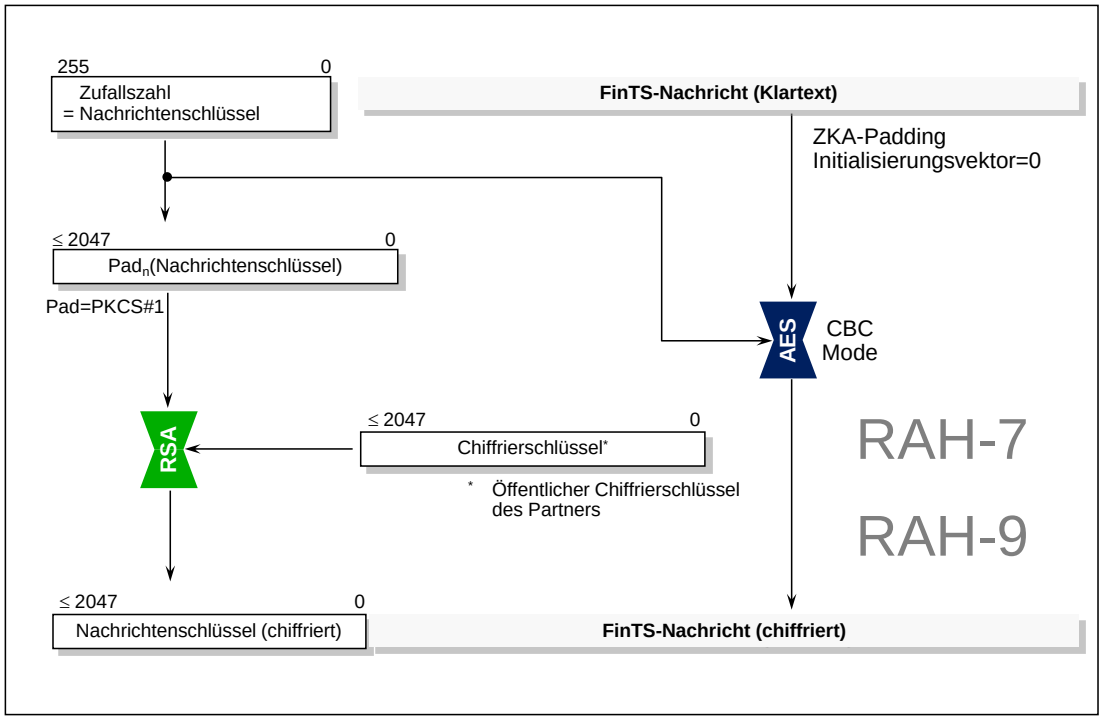


Abbildung 2: Verschlüsselung bei RAH-7 und RAH-9

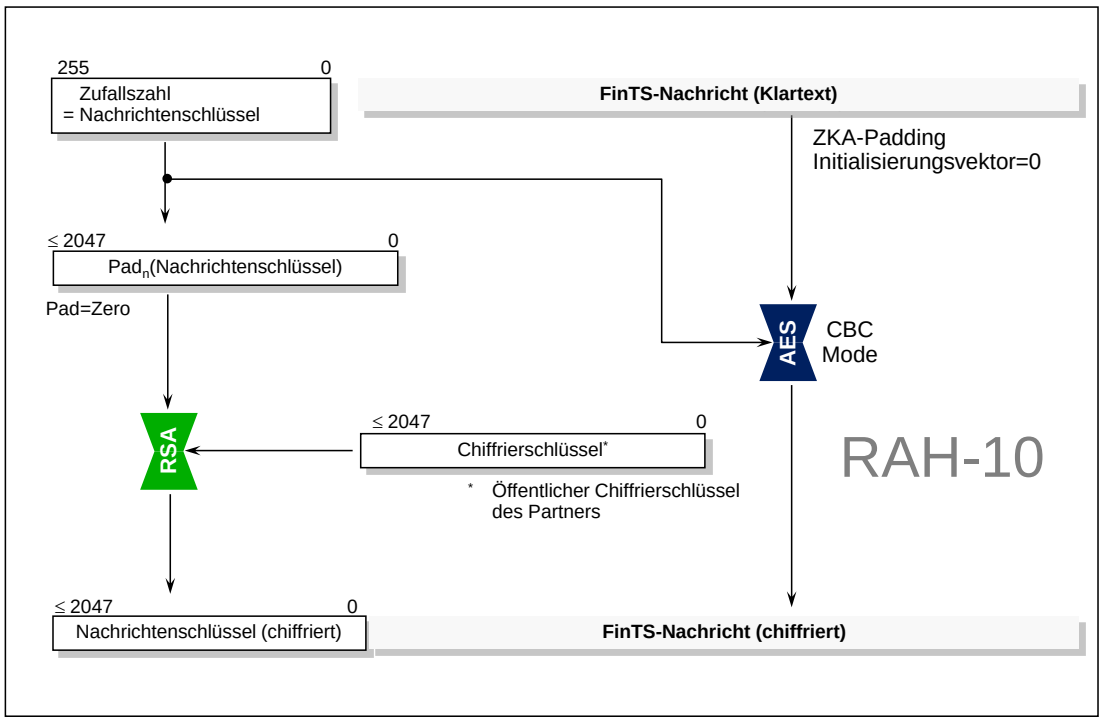


Abbildung 3: Verschlüsselung bei RAH-10

II.2.3 Komprimierung

Die Komprimierung wird unabhängig vom Sicherheitsprofil nach dem deflate- oder auch GZIP-Algorithmus gemäß [RFC 1951] vorgenommen. Andere Algorithmen können zusätzlich optional angeboten werden. Zum deflate-Algorithmus gibt es eine freie, auch in kommerziellen Produkten einsetzbare Referenzimplementierung so-

wohl in Source-Form als auch als binäre Bibliothek für alle gängigen Plattformen (http://www.gzip.org/zlib).

Das Kreditinstitut darf nur komprimiert antworten, wenn das Kundensystem (z. B. ein Smartphone) auch komprimiert gesendet hat. Damit wird vermieden, dass ein Kundensystem eine komprimierte Nachricht erhält und diese ggf. nicht verarbeiten kann.

II.2.4 Sicherheitsmedien beim Kundensystem

Als Sicherheitsmedien können Bankensignaturkarten oder Softwareschlüssel verwendet werden.

Bei Verwendung einer vom Kreditinstitut ausgegebenen Signaturkarte muss die Berechnung der kryptographischen Funktionen so durchgeführt werden, dass die kartenindividuellen Schlüssel niemals die Chipkarte verlassen. Es wird die in *IV.1 Chipapplikation für* beschriebene Bankensignaturkarte empfohlen.

Auf dem Sicherheitsmedium wird unter anderem der private Schlüssel des Benutzers gespeichert. Es ist aber auch möglich, öffentliche Schlüssel des Kreditinstitutes darauf abzulegen oder aber im Falle einer Chipkarte die kryptographischen Operationen damit durchzuführen. Generell müssen die geheimen Daten (z. B. private Schlüssel, Passworte) gegen unberechtigtes Auslesen geschützt sein.

Es ist zwingend erforderlich, die Daten auf dem Sicherheitsmedium (kryptographisch) zu schützen. Speziell ist im Rahmen der Speicherung von Softwareschlüsseln sicherzustellen, dass die Daten unter Einbeziehung eines vom Kunden frei wählbaren Kennworts verschlüsselt werden und der Zugriff auf die verschlüsselten Daten nur über die manuelle Eingabe des entsprechenden Kennwortes möglich ist.

II.3 Abläufe

II.3.1 Schlüsselverwaltung

♦ **Schlüsselarten**

Bei den Sicherheitsverfahren RAH-9 und RAH-10 können Benutzer und Kreditinstitut über zwei Schlüssel bzw. Schlüsselpaare verfügen:

- ein Signierschlüsselpaar
- ein Chiffrierschlüsselpaar

Der Signierschlüssel wird zum Unterzeichnen von Transaktionen verwendet, während der Chiffrierschlüssel zum Verschlüsseln von Nachrichten dient.

Bei dem Verfahren RAH-7 verfügt das Kreditinstitut ebenfalls nur über die obigen beiden Schlüssel bzw. Schlüsselpaare. Benutzer können jedoch über bis zu drei Schlüssel bzw. Schlüsselpaare verfügen:

- ein Schlüsselpaar für digitale Signaturen (DS-Schlüssel)
- ein Signierschlüsselpaar (Authentifikation)
- ein Chiffrierschlüsselpaar

Abhängig von der Personalisierung der Chipkarte können Signier- und Chiffrierschlüsselpaare identisch sein.

Der Signierschlüssel und der DS-Schlüssel werden zum Unterzeichnen von Transaktionen verwendet, während der Chiffrierschlüssel zum Verschlüsseln von Nachrichten dient. Falls kreditinstitutsseitig nur Geschäftsvorfälle angeboten werden, für die gemäß Bankparameterdaten die Unterzeichnung mit dem Signierschlüssel ausreichend ist, ist der DS-Schlüssel nicht erforderlich.

Bei Verwendung von Softwareschlüsseln gemäß Sicherheitsprofil RAH-10 wird dringend empfohlen, dass getrennte Signier- und Chiffrierschlüsselpaare zum Einsatz kommen.

♦ **Schlüsselnamen**

Der Schlüsselname setzt sich aus den folgenden alphanumerischen Komponenten zusammen:

- Ländercode
  (max. 3 Byte, es wird gemäß ISO 3166 der numerische Ländercode verwendet)
- Kreditinstitut
  (max. 30 Byte, normalerweise Kreditinstitutscode (Bankleitzahl)
- Benutzerkennung (nur für Benutzerschlüssel)
  (max. 30 Byte, kann vom Kreditinstitut festgelegt werden, vgl. [Formals], Abschnitt *II.1.1 Nachrichtenelemente*)
- Schlüsselart
  (1 Byte, D: DS-Schlüssel (nur Benutzerschlüssel bei RAH-7); S: Signierschlüssel; C: Chiffrierschlüssel)

- Schlüsselnummer
  (max. 3 Byte, numerisch)
- Versionsnummer
  (max. 3 Byte, numerisch)

Falls kein öffentlicher Schlüssel des Kreditinstituts vorliegt, so sind Schlüsselnummer und -version wegzulassen. Damit wird kreditinstitutsseitig auf den aktuell gültigen Schlüssel referenziert (Ein Kreditinstitut kann während einer Übergangszeit evtl. mehrere Schlüssel bis zu einem Verfallsdatum vorhalten. Aktuell gültig ist jeweils der neueste Schlüssel).

♦ **Generierung von Einmalschlüsseln (Nachrichtenschlüsseln)**

Zur Chiffrierung von Nachrichten wird ein dynamisch erzeugter Einmalschlüssel verwendet, der durch Generieren einer 32 Byte langen Zufallszahl gebildet wird.

♦ **Schlüsselgenerierung bei RAH**

Die Schlüsselpaare des Benutzers sind vom Kundensystem bzw. von der Chipkarte zu erzeugen. Die Schlüsselpaare des Kreditinstituts sind vom Kreditinstitut zu erzeugen. Die privaten Schlüssel sind jeweils geheim zu halten.

Die Schlüsselgenerierung hat gemäß dem folgenden Ablauf stattzufinden:[1]

1. Es wird ein konstanter öffentlicher Exponent e und ein für jeden Benutzer individueller Modulus n für jedes eingesetzte RSA-Schlüsselsystem verwendet.
2. Der konstante öffentliche Exponent e wird auf die 4. Fermat'sche Primzahl festgelegt: $e = 2^{16} + 1$
3. Der Modulus n eines jeden RSA-Schlüsselsystems hat eine Länge von N Bit. Es sind keine führenden 0-Bits erlaubt, so dass auf jeden Fall gilt: $2^{N-1} \leq n < 2^N$
4. Der Zielwert für N ist bei RAH-7, RAH-9 und RAH-10 $\leq$ 2047 Bit.

Schlüsselgenerierung bei RAH10:
Das Kundensystem muss sicherstellen, dass die Schlüssellänge eines neu generierten Schlüsselpaares des Kunden gleich der Länge des öffentlichen Signierschlüssels des Instituts ist, falls das Institut Institutssignaturen unterstützt. Anderenfalls ist die Länge des Chiffrierschlüssels maßgebend.

II.3.1.1.1 Behandlung von Zertifikaten

In FinTS ist die Verwendung von Zertifikaten durch die vorgesehenen Elemente unterstützt, es existieren jedoch keine Prozesse für das Zertifikatsmanagement. Diese sollen zu einem späteren Zeitpunkt auf Basis einer standardisierten Zertifizierungsinfrastruktur übernommen werden.

---

[1] Das Verfahren entspricht dem des DFÜ-Abkommens.

Folgende Festlegungen gelten für die Belegung der Zertifikatsfelder in den FinTS-Segmenten:

1. Allgemein
   Bei Verwendung des Signaturschlüssels (DS-Schlüssel) wird grundsätzlich in allen Nachrichten ein Zertifikat im Signaturkopf mitgeschickt.

   Bei Verwendung des Authentifikationsschlüssels (S-Schlüssel) kann ein Zertifikat in den Signaturkopf eingestellt werden.
   Im Verschlüsselungskopf kann ebenfalls ein Zertifikat eingestellt werden.
   Ggf. dort eingestellte Zertifikate können vom Institut ignoriert werden.

2. Erstmalige Übermittlung Kundenschlüssel bzw. Schlüsseländerung
   Bei der erstmaligen Übermittlung der Kundenschlüssel bzw. bei der Schlüsseländerung wird grundsätzlich der Authentifikationsschlüssel (S-Schlüssel) und wahlweise das zugehörige Zertifikat verwendet. Das Zertifikat wird nur in das vorgesehene Element im Geschäftsvorfall (HKSAK bzw. HKISA) eingestellt (nicht in den Signaturkopf).

3. Signaturkarten-Profil mit drei unterschiedlichen Schlüsseln
   Wenn ein Signaturkarten-Profil mit 3 unterschiedlichen Schlüsseln verwendet wird, muss bei der erstmaligen Übermittlung der Kundenschlüssel bzw. der Schlüsseländerung auch die Möglichkeit bestehen, das Zertifikat für den eigenen Verschlüsselungsschlüssel mitzuschicken.

♦ **Ausgewählte Beispiele für Rückmeldungscodes**

Code	Beispiel
9351	Zertifikat noch nicht gültig
9352	Zertifikat zurückgezogen bzw. gesperrt
9353	Zertifikatssignatur falsch
9354	Zertifizierungsinstanz (Herausgeber) nicht akzeptiert
9355	Fehler im Zertifikatsaufbau
9356	Zertifikatstyp nicht akzeptiert

II.3.1.1.2 Initiale Schlüsselverteilung

Der Benutzer benötigt für das Einrichten eines neuen Zugangs folgende Initialinformationen:

- seine Benutzerkennung
- Informationen zum Kommunikationszugang

Die Übermittlung dieser Informationen ist auf folgenden Wegen denkbar:

- Schriftstück des Kreditinstitutes (Benutzerkennung und Zugangsdaten müssen manuell vom Benutzer eingegeben werden)
- Softwareschlüssel z. B. auf einem USB-Stick des Kreditinstitutes mit folgendem Inhalt:

- allgemeiner Teil der UPD inkl. Benutzerkennung
- BPD oder BPD-Ausschnitt (Zugangsdaten-Segment) mit den Kommunikationszugangsdaten des jeweiligen Instituts

- Chipkarte des Kreditinstitutes, die die Kommunikationszugangsdaten in der Applikation EF_NOTEPAD enthält.

Zu Beginn muss ein gegenseitiger Austausch der öffentlichen Schlüssel von Benutzer und Kreditinstitut erfolgen. Alternativ erfolgt dieser Austausch durch eine Anforderung der Zertifikate bei den jeweiligen Zertifizierungsinstanzen. Dieser Prozess findet außerhalb des FinTS-Protokolls statt und wird daher hier nicht näher beschrieben.

Erfolgt der Schlüsselaustausch im Rahmen eines FinTS-Dialoges ist hierzu folgender Ablauf vorgesehen:

♦ **Initiale Schlüsselverteilung des Kreditinstituts**

1. Das Kreditinstitut übermittelt seinen öffentlichen Chiffrierschlüssel an den Benutzer. Falls es Nachrichten signiert, übermittelt es ebenfalls seinen öffentlichen Signierschlüssel. Hierzu gibt es zwei Möglichkeiten:
   - Zusenden bzw. Aushändigung der Schlüssel und anderer relevanter Daten auf einer Chipkarte z. B. bei Vertragseröffnung.
   - Übertragung der Schlüssel beim Erstzugang (z. B. bei Softwareschlüsseln)

     (1) Der Benutzer fordert die öffentlichen Schlüssel und die BPD mit Hilfe der Key-Management-Nachricht „Erstmalige Anforderung der Schlüssel des Kreditinstituts“ (s. *II.6.1.2 Erstmalige Anforderung der Schlüssel des Kreditinstituts*) an. Diese Nachricht ist weder signiert noch chiffriert.

     (2) Der weitere Ablauf ist abhängig davon, ob das Kreditinstitut seine Antwortnachrichten signiert.

     Fall A: Das Kreditinstitut signiert

     Der Benutzer erhält die öffentlichen Schlüssel des Kreditinstituts zurückgemeldet. Während die Authentizität des Chiffrierschlüssels dabei durch die Signatur gesichert ist, ist die Authentizität des Signierschlüssels nicht gesichert, da das Kundensystem die Echtheit der Signatur noch nicht prüfen kann.

     Fall B: Das Kreditinstitut signiert nicht

     Der Benutzer erhält nur den öffentlichen Chiffrierschlüssel zurückgemeldet. Dessen Authentizität ist dabei nicht gesichert.

     (3) Die Sicherung der Authentizität dieser Schlüssel kann über folgende Mechanismen erfolgen:

     Fall A: Ini-Brief Kunde → Kreditinstitut

     Diese Nachricht wird von einem Ini-Brief an den Benutzer begleitet. Die Gestaltung ist dem Kreditinstitut freigestellt, sollte sich aber am Muster in *Abbildung 5: Beispiel für die Gestaltung des Ini-Briefs* orientieren. Der Ini-Brief enthält für den Fall A Exponent und Modulus des Signierschlüssels sowie dessen Hash-Wert und für den Fall B Exponent und Modulus des Chiffrierschlüssels so-

wie dessen Hash-Wert.

Exponent und Modulus sind dabei mit führenden Nullen (X'00') auf die reale Länge des Modulus zu ergänzen. Ferner enthält der Ini-Brief den jeweiligen Schlüsselnamen.

Als Hashwertverfahren ist derzeit ausschließlich SHA-256 vorgesehen.

Bei der Hash-Wertbildung ist wie folgt vorzugehen:

a) Padding der höchstwertigen Bits von Exponent und Modulus des Schlüssels mit Nullen (X'00') auf den Zielwert der Schlüssellänge

b) Konkatenierung von Exponent und Modulus (Exponent || Modulus)

c) Bildung des Hash-Wertes mittels SHA-256 gemäß *II.2.1.1 Hashing)* über diesen Ausdruck

Nach Erhalt des Ini-Briefs führt der Benutzer einen Vergleich des im Ini-Brief aufgeführten Hash-Wertes mit dem Hash-Wert des vom Kreditinstitut übermittelten Schlüssels durch.

Bei Übereinstimmung der Hash-Werte gelten der bzw. die öffentlichen Schlüssel des Kreditinstituts als authentisiert.

Das Kundensystem sollte den Hash-Wert-Vergleich für den Benutzer in geeigneter Weise unterstützen.

Fall B: Übermittlung des Hash-Wertes auf der Chipkarte

Auf der Karte befindet sich in der Applikation EF_NOTEPAD (s. *IV.1.1 Applikation Notepad*) für Fall A der Hash-Wert des öffentlichen Signierschlüssels des Kreditinstituts und für Fall B der Hash-Wert des öffentlichen Chiffrierschlüssels des Kreditinstituts. Die Hash-Wert-Bildung erfolgt wie in Fall A.

Dieser Hash-Wert wird vom Kundensystem mit dem Hash-Wert des in der Nachricht übermittelten Schlüssels verglichen.

Das Kundensystem sollte den Benutzer über das Ergebnis des Hash-Wertvergleichs informieren.

Bei Übereinstimmung der Hash-Werte gelten der bzw. die öffentlichen Schlüssel des Kreditinstituts als authentisiert.

Fall C: Prüfung des übermittelten Zertifikates

Falls das Kreditinstitut über zertifikatsbasierte Schlüssel verfügt, übermittelt es das jeweilige Zertifikat in der Nachricht zusammen mit dem öffentlichen Schlüssel. Somit ist der Benutzer in der Lage, das Zertifikat bei der jeweiligen Zertifizierungsinstanz zu verifizieren. Diese Verifikation findet außerhalb des FinTS-Protokolls statt und wird daher hier nicht näher beschrieben.

Ein Hash-Wert-Vergleich wie in den beiden anderen Fällen ist nicht erforderlich.

Das Kundensystem sollte den Benutzer über das Ergebnis der Zertifikatsprüfung informieren.

♦ **Initiale Schlüsselverteilung des Kundensystems**

2. Der Benutzer übermittelt alle seine öffentlichen Schlüssel, die mit dem privaten Signierschlüssel unterzeichnet wurden, im Rahmen der Key-Management-Nachricht „Erstmalige Übermittlung der Schlüssel des Benutzers“ an das Kreditinstitut (vgl. *II.6.1.3 Erstmalige Übermittlung der Schlüssel des Benutzers*). Diese Nachricht muss sowohl signiert als auch chiffriert sein.
3. Um die Authentizität der Schlüssel zu gewährleisten, sind folgende Mechanismen möglich:

   Fall A: Ini-Brief Kreditinstitut → Kunde

   - Der Benutzer erfährt anhand des Rückmeldungscodes 3310 („Ini-Brief erforderlich“) in der Kreditinstitutsnachricht, dass diese Nachricht durch einen Ini-Brief gemäß dem in *Abbildung 5: Beispiel für die Gestaltung des Ini-Briefs bei RAH-9 bzw. RAH-10* aufgeführten Muster begleitet werden muss. Im Ini-Brief bestätigt der Benutzer ausschließlich den öffentlichen Signierschlüssel mit handschriftlicher Unterschrift. Eine Bestätigung des öffentlichen Chiffrierschlüssels ist nicht erforderlich, da dieser mit dem Signierschlüssel signiert wird und damit authentifiziert ist. Neben dem Schlüssel und dem Schlüsselnamen wird im Ini-Brief der Hash-Wert des Schlüssels aufgeführt. Dieser wird ebenso gebildet wie der Hashwert im Ini-Brief des Kreditinstituts (s.o.).
   - Im Kreditinstitut findet ein Vergleich zwischen dem im Ini-Brief aufgeführten Hash-Wert und dem Hash-Wert des vom Benutzer übermittelten öffentlichen Signierschlüssels statt. Falls dieser Vergleich positiv verläuft, werden die öffentlichen Schlüssel des Benutzers freigeschaltet.

   Fall B: Prüfung des übermittelten Zertifikates

   Der Benutzer erfährt anhand des Rückmeldungscodes 3320 („Ini-Brief nicht erforderlich“) in der Kreditinstitutsnachricht, dass das Kreditinstitut die Prüfung der Authentizität der Schlüssel auf Basis eines Zertifikates vornehmen kann.

   Falls der Benutzer über zertifikatsbasierte Schlüssel verfügt, übermittelt er daher das jeweilige Zertifikat in der Nachricht zusammen mit dem öffentlichen Schlüssel. Somit ist das Kreditinstitut in der Lage, das Zertifikat bei der jeweiligen Zertifizierungsinstanz zu verifizieren. Diese Verifikation findet außerhalb des FinTS-Protokolls statt und wird daher hier nicht näher beschrieben.

   Im nächsten Schritt werden die öffentlichen Schlüssel des Benutzers freigeschaltet. Ein Hash-Wert-Vergleich wie in Fall A ist nicht erforderlich.
4. .Bei RAH-10 hat eine Synchronisierung der Kundensystemkennung zu erfolgen (s. [Formals], Abschnitt *III.3 Synchronisierung*).

5. Nachdem die Erstinitialisierung abgeschlossen ist, kann der Benutzer Auftragsnachrichten senden.

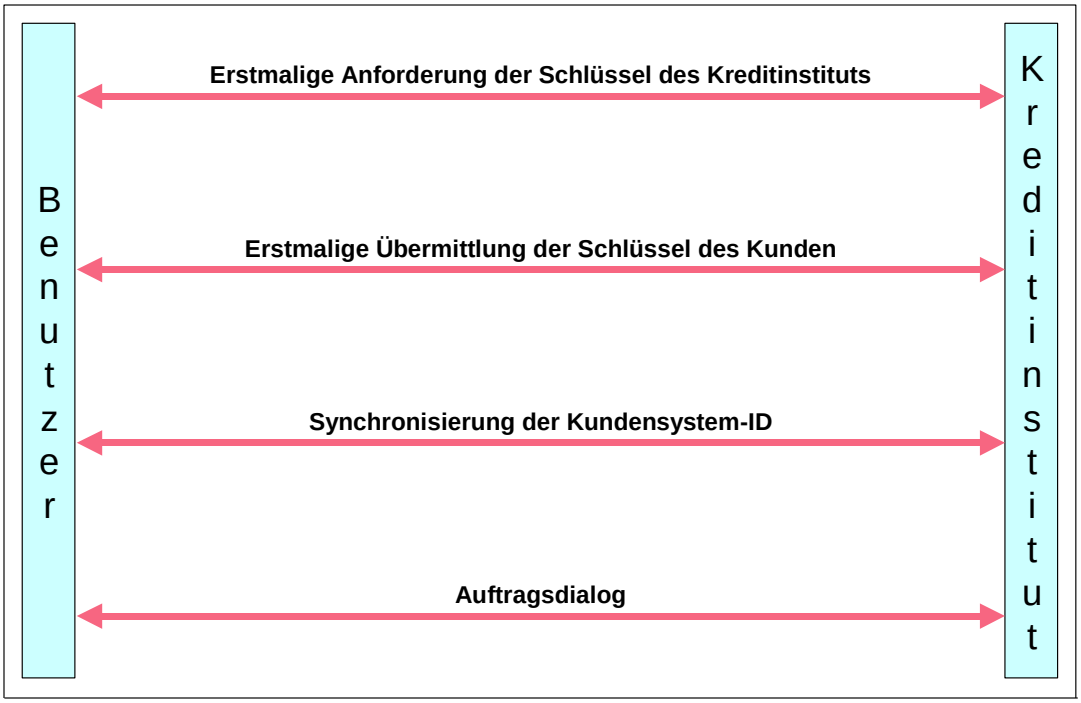


Abbildung 4: Ablauf der Erstinitialisierung bei RAH

Um die Multibankfähigkeit verschiedener Kundensysteme zu sichern, gelten für die Ini-Datei z. B auf einem USB-Stick folgende Namenskonventionen:

- UPD allgemeiner Teil: <Benutzerkennung>.UPA
- Datei mit den öffentlichen Schlüsseln: <Benutzerkennung>.PKD
- BPD: <Bankleitzahl>.BPD
- Segment mit Kommunikationszugang: <Bankleitzahl>.KOM

Falls die Benutzerkennung nicht im Dateisystem darstellbar ist, ist sie entsprechend zu kürzen. Der USB-Stick muss im Standardformat des jeweiligen Betriebssystems formatiert sein. Die Dateien sind im Stammverzeichnis des USB-Stick abzulegen.

Die folgende Abbildung zeigt ein Beispiel für einen Ini-Brief bei Verwendung des Sicherheitsprofils RAH-9 und RAH-10 mit SHA-256 als Hashwertverfahren.

Ini-Brief HBCI

Benutzername	______________	Kundensoftware-interner Name (Angabe freigestellt)
Datum	______________	Datum der Erstellung des Initialisierungsauftrags (TT.MM.JJJJ)
Uhrzeit	______________	Uhrzeit der Erstellung des Initialisierungsauftrags (hh:mm)
Empfänger	______________	Kreditinstitutskennung (wird vom jeweiligen Kreditinstitut mitgeteilt)
Benutzerkennung	______________	max. 30 Stellen alphanumerisch (wird vom jeweiligen Kreditinstitut mitgeteilt)
Schlüsselnummer	______________	Nummer des Signierschlüssels (max. 3 Stellen)
Schlüsselversion	______________	Version des Signierschlüssels (max. 3 Stellen)
HBCI-Version	______________	derzeit 4.1
Sicherheitsprofil	______________	RAH-7, RAH-9, oder RAH-10

Öffentlicher Schlüssel für die elektronische Signatur:

Exponent <Reale Moduluslänge>

```
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  01  00  01
```

Modulus ..2048

```
97  A9  D8  FD  68  46  BA  8C  5E  DB  E0  F5  1C  35  41  41
3C  4A  B3  7C  95  6A  CC  11  29  E3  A7  63  A6  7B  A8  6D
E8  1D  99  D2  74  24  6E  6C  C6  FD  A9  9C  A2  19  15  44
BF  2A  5D  B6  93  5B  FA  FB  88  19  FE  B2  1C  96  FB  4F
5E  66  6A  2D  F8  80  5E  84  9B  42  8C  E6  9A  76  AB  52
5A  36  C1  6D  81  D1  65  0A  BC  86  35  86  E7  8E  5B  B5
EF  E0  84  A1  44  76  69  DE  34  D5  31  A2  D0  25  02  78
B5  BE  AB  C0  D9  30  B1  04  91  DF  0D  3B  E1  33  3D  E7
24  57  CD  E5  A2  13  EF  95  4B  23  81  58  91  60  85  71
39  3E  00  92  7C  F9  06  3A  97  3D  3F  87  52  80  AF  4E
37  39  F1  9E  14  A5  71  30  33  6C  E4  DD  38  66  E8  B2
29  6B  BF  F2  25  36  3E  0F  A0  55  39  4A  31  7A  69  88
9B  D6  26  B6  B6  53  CA  C7  40  4A  60  79  29  C4  97  DC
43  3B  54  A6  B6  44  33  E0  E8  05  17  43  50  13  EE  D4
F2  EF  F2  BD  B2  2A  B9  87  52  95  FC  56  93  24  75  A7
E7  D1  CD  90  9A  D3  D8  78  99  1D  C3  21  4D  2F  5A  A3
```

Hashwert (SHA-256):

```
BF  C2  A1  01  7C  65  31  9C  8B  1B  D8  26  09  85  E6  1F
A9  99  1A  65  17  BF  67  86  17  D8  7C  EE  DC  C3  61  11
```

Ich bestätige hiermit den obigen öffentlichen Schlüssel für meine elektronische Signatur.

_______________	______________________	______________________
Ort / Datum	Firma/Name	Unterschrift

Abbildung 5: Beispiel für die Gestaltung des Ini-Briefs bei RAH-9 bzw. RAH-10

II.3.1.1.3 Schlüsseländerungen

- **Routinemäßige Schlüsseländerung des Benutzers**

  Bei Speicherung der Schlüssel auf einer Chipkarte ist i. d. R. auf elektronische Weise keine Änderung einzelner kartenindividueller Schlüssel möglich. Im Falle einer routinemäßigen Schlüsseländerung (z. B. bei Ablauf des Zertifikates) oder einer vermuteten Kompromittierung muss daher ein Kartenaustausch oder ein Ersatz aller Schlüssel erfolgen.

  Falls die Karte die Generierung neuer Schlüssel zulässt oder im Falle anderer Speichermedien (z. B. Diskette oder USB-Stick) ändert der Benutzer seine Schlüsselpaare unabhängig voneinander.

  Der Benutzer sendet je Kreditinstitut im Rahmen einer FinTS- Kommunikation eine Nachricht, in welcher dieses über einen neuen öffentlichen Schlüssel informiert wird (vgl. *II.6.1.1 Änderung eines öffentlichen Schlüssels des Benutzers*). Die Nachricht ist mit dem alten (bei Wechsel des Signierschlüssels), respektive dem aktuellen (bei Wechsel des Chiffrierschlüssels) privaten Signierschlüssel des Benutzers zu signieren und mit dem aktuellen Chiffrierschlüssel des Kreditinstituts zu chiffrieren.

  Das Kreditinstitut speichert diesen neuen öffentlichen Schlüssel des Benutzers und verwendet ihn ab sofort (d. h. bereits in der Antwortnachricht) für alle Verschlüsselungen bzw. Verifikationen von Signaturen. Gleichzeitig kann der alte Schlüssel gesperrt werden. Zusätzlich ist es jedoch bei kartengestützten Verfahren – unabhängig von der Nutzung von Zertifikaten – erlaubt, einen Schlüssel für die Laufzeit der Karte weiter aktiv zu halten und somit zwei Schlüssel parallel zu unterstützen.

  Falls die Übermittlung der neuen Schlüssel aus irgendeinem Grunde fehlschlägt, kann der Benutzer den Vorgang beliebig wiederholen.

  Bei einer Schlüsseländerung wird die Signatur-ID auf 1 zurückgesetzt. Die Liste der eingereichten bzw. noch nicht eingereichten Signatur-IDs (s. *II.4 Bankfachliche Anforderungen*) wird gelöscht.

- **Routinemäßige Schlüsseländerung des Kreditinstituts**

  Ein Kreditinstitut generiert bei Bedarf ein neues Schlüsselpaar.

  Falls das Kreditinstitut über aktuellere öffentliche Schlüssel verfügt als der Bote der Initialisierungsnachricht, werden diese in der Kreditinstitutsantwort auf die Initialisierungsnachricht mit übertragen (vgl. [Formals], Abschnitt *II.15 Initialisierung*). Die neuen Schlüssel gelten ab sofort, d. h. bereits für die erste Folgenachricht. Da das Kreditinstitut i. d. R. aber auch noch die alten Schlüssel aktiv hält, werden für einen begrenzten Zeitraum auch noch Nachrichten akzeptiert, die mit den alten Kreditinstitutsschlüsseln chiffriert wurden.

  Zur Verifikation des kreditinstitutsseitigen öffentlichen Schlüssels auf dem Kundensystem kann das entsprechende Kreditinstitut die Kreditinstitutsnachricht mit dem alten Signierschlüssel signieren (wenn eine kreditinstitutsseitige Signatur vorgesehen ist) oder den Hash-Wert des öffentlichen Schlüssels analog der initialen Schlüsselverteilung an den Benutzer übermitteln. Die Verifikation ist grundsätzlich optional.

  Für den Fall, dass der alte Kreditinstitutsschlüssel nicht mehr zur Verfügung steht oder gesperrt werden musste, wird dem Benutzer – falls er den alten Kreditinstitutsschlüssel zur Chiffrierung der Initialisierung verwendet – der Rückmeldungscode "9030" mit dem Hinweis "Fehler beim Entschlüsseln" gesendet. Ggf. kann die Initialisierung vom Kreditinstitutssystem auch gar nicht verarbeitet werden, so dass keine Antwort gesendet wird. Daraufhin sollte das Kundensystem über eine anonymen

Kommunikation mit Hilfe der Nachricht „Erstmalige Anforderung der Schlüssel des Kreditinstituts“ (s. *II.6.1.2 Erstmalige Anforderung der Schlüssel des Kreditinstituts*) die neuen Kreditinstitutsschlüssel anfordern. Zur Verifikation der neuen Schlüssel muss dem Benutzer in diesem Fall zusätzlich ein Ini-Brief mit dem Hash-Wert des neuen Kreditinstitutsschlüssels zugeschickt werden.

II.3.1.1.4 Schlüsselverteilung nach Kompromittierung

Die Verteilung der Schlüssel nach einer Kompromittierung erfolgt analog der Schlüsselverteilung bei der Initialisierung. Es findet immer ein Austausch aller Schlüssel statt, auch dann, wenn nur einer der Schlüssel kompromittiert wurde.

II.3.2 Schlüsselsperrung

Bei der Schlüssel- bzw. Benutzersperrung muss zwischen folgenden Fällen unterschieden werden:

- Kompromittierung des eigenen Schlüssels
- Verlust des eigenen Schlüssels
- Überschreiten der Anzahl der Falschsignaturen

Zusätzlich müssen bei der Sperrung noch folgende Punkte berücksichtigt werden:

- Information des Benutzers
- Entsperrung

Die Sperrung anderer Benutzer wird als eigenständiger Auftrag behandelt und zu einem späteren Zeitpunkt realisiert.

♦ **Kompromittierung des eigenen Schlüssels**

Bei Verdacht auf Kompromittierung des eigenen Schlüssels kann die Sperrung mittels einer speziellen Nachricht (vgl. *II.6.1.4 Schlüsselsperrung durch den Benutzer*) erfolgen, welche signiert sein muss.

♦ **Verlust des eigenen Schlüssels**

Bei einem Verlust (inkl. Diebstahl) des eigenen Schlüssels (respektive des Speichermediums) muss der Benutzer Schlüssel bzw. Medium sperren und beim Kreditinstitut ein anderes Medium inkl. Schlüssel beantragen.

Eine nicht-signierungspflichtige Sperrmöglichkeit ist nicht vorgesehen, da hierdurch die Gefahr des Missbrauchspotential gegeben ist (absichtliche Sperrung fremder Anschlüsse).

Eine Sperrung auf anderem Weg (z. B. telefonische Sperrung über Servicezentralen) muss immer möglich sein (z. B. Verlust der eigenen Infrastruktur).

♦ **Überschreiten der Anzahl der Falschsignaturen**

Wird beim Einreichen von Aufträgen durch fehlerhafte Signaturen die festgelegte Anzahl von n Falschsignaturen in Folge überschritten, werden kreditinstitutsseitig die Schlüssel gesperrt. Als Falschsignaturen werden dabei fehlgeschlagene kryptographische Operationen, jedoch z. B. keine fehlerhaften Berechtigungen verstanden.

Bei einer Sperrung aufgrund zu vieler Fehlsignaturen werden alle Benutzerschlüssel gesperrt. Sofern die Nachricht lediglich von einem einzigen Benutzer signiert wurde oder falls bei einer mehrfach signierten Nachricht der Bote von der Fehlsignaturen-

sperre betroffen ist, wird die Kommunikation beendet. Der Kommunikationsabbruch erfolgt dabei kreditinstitutsseitig im Anschluss an die Antwortnachricht, d. h. ein Austausch von Endenachrichten findet nicht statt. Die Antwort ist signiert (sofern kreditinstitutsseitig signiert wird) aber nicht verschlüsselt. In der Antwortnachricht teilt das Kreditinstitut lediglich den Grund des Kommunikationsendes mit. Antworten auf Aufträge dürfen nicht mitgesendet werden, da diese aufgrund der Sperrung nicht abgesichert werden können.

- **Information des Benutzers**

  Im Falle einer Sperrung aufgrund von Schlüsselkompromittierung oder Schlüsselverlust erhält der Benutzer auf die Sperrnachricht eine Antwortnachricht (vgl. *II.6.1.4 Schlüsselsperrung durch den Benutzer*), welche ihm die Sperrung bestätigt. Bei einer Sperrung wegen Überschreitung des Maximalwertes möglicher Falschsignaturen erhält er lediglich einen entsprechenden Rückmeldungscode. In jedem Fall erhält er jedoch entsprechende Fehlermeldungen bei der Einreichung nachfolgender Nachrichten.

- **Entsperrung der Benutzerkennung**

  Eine Entsperrung erfolgt nur gegen handschriftliche Unterschrift des Benutzers.

  Ist der Schlüssel kompromittiert oder nicht mehr auffindbar, so wird für den Benutzer eine neue Chipkarte, respektive neue Schlüssel und ein neues EF_ID oder ein neues Schlüsselpaar (RAH) erzeugt und der alte Schlüssel bleibt gesperrt. Es werden in jedem Falle alle Schlüsselpaare neu vergeben, auch wenn nur ein Schlüsselpaar kompromittiert sein sollte. Damit ein Benutzer nach einer Sperrung wieder zum Zugang zum System autorisiert werden kann, darf er in diesem Fall ausnahmsweise einer erneute Erstinitialisierung durchführen und seine Schlüssel über einen Ini-Brief freischalten lassen.

  In den übrigen Fällen kann der Schlüssel einfach durch das Kreditinstitut entsperrt werden.

II.4 Bankfachliche Anforderungen

- **Zu signierende Nachrichten**

  Grundsätzlich sind alle Benutzernachrichten mindestens mit Botensignatur zu signieren, bei Sicherheitsprofil RAH-7 gemäß den in den BPD vorgegebenen Sicherheitsklassen. Ausnahmen gelten für den anonymen Zugang ([Formals], Abschnitt *II.17 Anonymer Zugang*), für die Schlüsselanforderung im Rahmen der Erstinitialisierung (*II.6.1.1 Änderung eines öffentlichen Schlüssels des Benutzers*) und für die Lebendmeldung ([Formals], Abschnitt *III.6 Lebendmeldung in Dialogen*).Die Signatur von Kreditinstitutsnachrichten ist optional.

- **Doppeleinreichungskontrolle**

  Die Doppeleinreichungskontrolle wird mittels eines Zählers pro Signatur realisiert (Signatur-ID), dessen Inhalt jeweils in die Signatur(en) der Nachricht einfließt. Falls als Sicherheitsmedium keine Chipkarte verwendet wird, wird zur Doppeleinreichungskontrolle zusätzlich zur Signatur-ID die Kundensystemkennung benötigt.

  Bei der Doppeleinreichungskontrolle (Verhinderung von Replay-Attacken) ist zu berücksichtigen, dass die sequentiell erzeugten Referenznummern (=Signatur-IDs) beim Kreditinstitut nicht in derselben Reihenfolge eintreffen müssen, da diese benutzerseitig auch offline (d. h. zeitlich voneinander unabhängig) generiert werden können. Das Kreditinstitut muss deshalb sicherstellen, dass innerhalb eines bestimmten Zeitraums keine Sequenznummer mehrfach erscheint.

  Aus diesem Grund muss beim Kreditinstitut eine Liste mit den eingereichten (Positivliste) oder noch nicht eingereichten (Negativliste) Signatur-IDs geführt werden. Nach einer festgelegten Aufbewahrungsfrist wird eine Referenznummer nicht mehr akzeptiert. (Konkret wird ein Kreditinstitut eine Nachricht abweisen, welche länger als die vereinbarte Frist nach einer Nachricht mit höherer Signatur-ID eintrifft). Diese Liste muss je Signaturschlüsselpaar geführt werden, d. h., falls der Benutzer sowohl mit dem Signierschlüssel- als auch mit dem DS-Schlüssel unterschreibt, sind zwei Listen erforderlich.

- **Mehrfachsignaturen**

  Bei Mehrfachsignaturen ist die Reihenfolge der Signaturen bedeutungslos.

  Sind die Berechtigungsprofile mehrerer signierender Benutzer zueinander inkonsistent, so liegt es im Ermessen des Kreditinstituts, ob es die Nachricht annimmt oder ablehnt (Beispiel: Der Herausgeber eines Auftragsteils, für deren Aufträge drei Signaturen erforderlich sind, liefert nur eine zweite Signatur eines Benutzers mit, der über das Recht verfügt, die Aufträge alleine zu signieren).

  Ob es zulässig ist, dass bei Mehrfachsignaturen verschiedene Signaturverfahren eingesetzt werden, gibt das Kreditinstitut in den BPD im Segment „Sicherheitsverfahren“ (siehe [Formals], Abschnitt *IV.2.3 Sicherheitsverfahren*) an.

- **verteilte Signaturen**

  Falls ein Auftrag mehrere Signaturen benötigt, diese jedoch nicht sofort erstellt und innerhalb der gleichen Nachricht übermittelt werden können, so besteht die Möglichkeit, die Signaturen auch zeitlich und räumlich getrennt zu erstellen. Die zugehörigen Abläufe sind in [Formals], Abschnitt *III.8 Verteilte Signaturen* beschrieben. Die hier vorgestellten Geschäftsvorfälle sind in den BPD als zulässig zu hinterlegen, ebenfalls die Geschäftsvorfälle, die verteilt signiert werden sollen.

II.5 Formate für Signatur und Verschlüsselung

Für die Speicherung der Sicherheitsinformationen für die Signatur(en) werden dem XML-Signature-Standard entsprechend Signatur-Segmente in die bestehende Nachricht eingefügt. Diese Segmente enthalten jeweils XML-Referenzen auf die signierten Teile der Gesamtnachricht. Dies muss im Falle einer Botensignatur die Gesamtnachricht umfassen, oder kann sich im Falle einer Signatur durch Herausgeber und Zeugen von Aufträgen auf die entsprechenden Aufträge beschränken (siehe auch [Syntax]).

Bei der Verschlüsselung wird der zu verschlüsselnde Teil der Nachricht gegen ein nach dem XML-Encryption-Standard aufgebautes Verschlüsselungsdaten-Segment ausgetauscht, welches neben den für die Entschlüsselung benötigten Informationen auch die verschlüsselten Daten selbst enthält.

Weitere Informationen zum Signieren und Verschlüsseln sind unter [Formals], Abschnitt *II.12.4 Vorgehensweise beim Signieren und Verschlüsseln* zu finden.

Für die Übermittlung der sicherheitsrelevanten Informationen werden die folgenden Segmente und Datenelementgruppen übertragen.

II.5.1 Signatur-Segment

♦ **Beschreibung**

Das Signatur-Segment enthält den Signaturwert sowie Zusatzinformationen, die allgemein für alle im FinTS-Protokoll verwendbaren Sicherheitsverfahren benötigt werden, oder die für das jeweilige Sicherheitsverfahren spezifisch sind. Je nach verwendeter Syntax können weitere syntaxspezifische Zusatzinformationen enthalten sein (z. B. Algorithmenbezeichner).

Je nach Erfordernissen der verwendeten Syntax werden im Signatursegment auch der Hashwert und Referenzen auf die signierten Nachrichtenteile abgelegt. Auch ist es möglich, dass die verwendete Syntax ein mehrstufiges Hashing realisiert, wobei unterschiedliche Teile der Nachricht einzeln gehasht und anschließend nochmals ein Gesamt-Hashwert gebildet wird. In diesem Fall werden alle Hashwerte im Signatur-Segment gespeichert und mitsigniert.

Bei Verwendung als Botensignatur wird der gesamte Inhalt der Nachricht einschließlich aller im Signatursegment enthaltenen Hashwerte, Referenzen und Zusatzinformationen signiert, ausgenommen sind lediglich der Signaturwert selbst sowie der Bezeichner des Signaturschlüssels.

Bei Verwendung als Signatur des Auftragsteils werden ein- oder mehrere Aufträge eines Auftragsteils sowie die Hashwerte, Referenzen und Signatur-Zusatzinformationen außer dem Signaturwert selbst sowie dem Bezeichner des Signaturschlüssels signiert.

Wenn Referenzen auf Nachrichtenteile gespeichert werden, ist zu gewährleisten, dass diese bei Signaturen eines Auftragsteils unabhängig vom Kontext der Gesamtnachricht bleiben.

Allgemeine Informationen:

- Rolle des Signierenden
- Sicherheitsdatum und –uhrzeit

Für HBCI-Sicherheit spezifische Informationen:

- die verwendeten Algorithmen (siehe *II.1*)
- Kennzeichen für Sicherheitsprofil (siehe *II.1*)
- Signatur-ID
- Schlüsselname bestehend aus:
    - Kreditinstitutskennung
    - Benutzerkennung (bei RAH-Benutzerschlüsseln)
    - Nummer
    - Version
    - Typ
- Zertifikat

- Kundensystemkennung (bei sofwarebasiertem RAH)
- Kartenidentifizierung (CID) der RSA-Chipkarte (bei kartenbasiertem RAH)
- Nummer und Version des dem Benutzer vorliegenden Bankschlüssels (bei Benutzersignaturen)

♦ **Belegungsrichtlinien**

Rolle des Signierenden

Die folgenden Codes können für die folgenden Signaturarten verwendet werden, sofern dies zwischen Benutzer und Kreditinstitut zuvor vereinbart wurde:

1. Botensignatur:

Es ist genau eine Botensignatur zu erstellen. Der Lieferant einer Botensignatur ist implizit immer Bote der Gesamtnachricht. Er kann darüber hinaus aber auch Herausgeber oder Zeuge eines darin enthaltenen Auftragsteils sein, welchen er in jedem Fall mitsigniert. Die folgenden Codes können mit den folgenden Bedeutungen verwendet werden:

- MSG: Der Sicherheitslieferant ist nur Bote der Gesamtnachricht.
- ISS: Der Sicherheitslieferant ist sowohl Bote der Gesamtnachricht als auch Herausgeber des darin enthaltenen Auftragsteils. Das Kreditinstitut signiert grundsätzlich in dieser Rolle.
- WIT: Der Sicherheitslieferant ist sowohl Bote der Gesamtnachricht als auch Zeuge des darin enthaltenen Auftragsteils.

2. Auftragssignatur:

In jedem Auftragsteil sind beliebig viele Signaturen über alle Aufträge des Auftragsteils oder eine Teilmenge davon möglich. Der Lieferant einer Auftragssignatur kann Herausgeber oder Zeuge der Aufträge sein, die er signiert hat. Die folgenden Codes können mit den folgenden Bedeutungen verwendet werden:

- ISS: Der Sicherheitslieferant ist Herausgeber der von ihm signierten Aufträge. Das Kreditinstitut signiert grundsätzlich in dieser Rolle.
- WIT: Der Sicherheitslieferant ist Zeuge der von ihm signierten Aufträge.

Kennzeichen für Sicherheitsprofil

Da das aktuelle Sicherheitsprofil aufgrund der Algorithmusbezeichner ggf. nicht eindeutig bestimmt werden kann, hat der Sicherheitslieferant hier zusätzlich ein Kennzeichen für das aktuell von ihm verwendete Sicherheitsprofil einzustellen.

Signatur-ID

Wert, der zur Doppeleinreichungskontrolle in die Signatur einzustellen ist.


Wenn eine Synchronisierung der Kundensystemkennung durchgeführt wird, wird die eingestellte Signatur-ID nicht überprüft, da eine Doppeleinreichungskontrolle für eine Synchronisierungsnachricht dieses Typs nicht notwendig ist. Der Benutzer kann hier somit einen beliebigen Wert einstellen (siehe auch [Formals], Abschnitt *III.2 Statusprotokoll*)

In Kreditinistutsnachrichten kann entweder der Wert aus der Auftragssignatur verwendet werden, oder das Institut führt einen eigenen ID-Zähler.

Kartenidentifikation

Bei kartenbasiertem RAH ist hier die CID der verwendeten Chip-Karte einzustellen.

Kundensystemkennung

Bei softwarebasiertem RAH ist hier eine zuvor vom Kreditinstitut angeforderte Kennung für das eigene Kundensystem einzustellen.


Wenn eine Synchronisierung der Kundensystemkennung durchgeführt wird, wird die eingestellte Kundensystemkennung nicht überprüft. Der Benutzer kann hier somit einen beliebigen Wert einstellen, wenn ihm keine echte Kundensystemkennung vorliegt (siehe auch [Formals], Abschnitt *III.2 Statusprotokoll*)

Nummer und Version des dem Kunden vorliegenden Bankschlüssels

Im Falle einer Signatur durch einen FinTS-Benutzer hat dieser in seiner Signatur auch die Schlüsselnummer und -version des ihm vorliegenden Bankschlüssels anzugeben. So ist für eine spätere Signatur durch das Kreditinstitut sichergestellt, mit welchen Schlüsseln diese Signatur zu erfolgen hat, damit der Benutzer sie verifizieren kann.

Im Falle von Kreditinstitutssignaturen sind die Felder leer zu lassen.

Zertifikat

Bei RAH-7 in Verbindung mit mindestens einem zu signierenden Geschäftsvorfall der Sicherheitsklasse 3 oder 4 ist ein Zertifikat Pflicht.

Bei RAH-7 und RAH-9 in Verbindung mit zu signierenden Geschäftsvorfällen der Sicherheitsklassen 1 bis 2 ist ein Zertifikat optional.

Details der syntaktischen Umsetzung finden sich in [Syntax].

II.5.2 Verschlüsselungsdaten

♦ **Beschreibung**

Für die Speicherung der Sicherheitsinformationen für die Verschlüsselung werden Verschlüsselungsdaten-Segmente in die bestehende Nachricht eingefügt. Sie ersetzen jeweils die Nachrichtenteile, die sie in verschlüsselter Form enthalten. Darüberhinaus enthalten sie Zusatzinformationen, die teilweise für das jeweilige Sicherheitsverfahren spezifisch sind sowie gegebenenfalls syntaxspezifische Zusatzinformationen (z. B. Algorithmusbezeichner).

Für HBCI-Sicherheit spezifische Informationen:

- Schlüsselname bestehend aus:
  - Kreditinstitutskennung
  - Benutzerkennung (bei RAH-Benutzerschlüsseln)
  - Nummer
  - Version
  - Typ
- die verwendeten Algorithmen (siehe *II.1.2*)
- Kennzeichen für Sicherheitsprofil (siehe *II.1.2*)

♦ **Belegungsrichtlinien**

Kennzeichen für Sicherheitsprofil

Da das aktuelle Sicherheitsprofil aufgrund der Algorithmusbezeichner ggf. nicht eindeutig bestimmt werden kann, hat der Sicherheitslieferant hier zusätzlich ein Kennzeichen für das aktuell von ihm verwendete Sicherheitsprofil einzustellen.

Details der syntaktischen Umsetzung finden sich in [Syntax].

II.5.3 Komprimierungsdaten

- **Beschreibung**

  Die Komprimierung wird formal als Verschlüsselung betrachtet (siehe *II.5.2*). Als Algorithmen sind ausschließlich die vom Kreditinstitut laut BPD unterstützten Komprimierungsalgorithmen zulässig.

  Details der syntaktischen Umsetzung mittels finden sich in [Syntax].

II.6 Key-Management

II.6.1 Key-Management-Nachrichten

Aufträge des Key-Managements dürfen nur in speziellen Nachrichten übertragen werden, siehe [Syntax].

Die Nachrichten für das Key-Management müssen zum Teil kryptographisch geschützt werden. Alternativ können auch Offline-Sicherungsverfahren (z. B. Brief) zum Einsatz kommen (vgl. ***Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.***). In den Beschreibungen ist jeweils angegeben, welche Verfahren anzuwenden sind.

Für alle kryptographisch zu signierenden Nachrichten gilt: es muss mindestens eine Signatur des Schlüsseleigentümers als Auftraggebersignatur (Signatur mit Rolle ISS) vorhanden sein. Bei direkter Einreichung der Nachricht (ohne Intermediär) bringt der Auftraggeber hierfür eine Botensignatur mit dem Rollenkennzeichen ISS an, eine zusätzliche Signatur im Auftragsteil ist optional. Bei Einreichung über einen Intermediär wird die Botensignatur durch den Intermediär geleistet, so dass die Auftraggebersignatur als Signatur im Auftragsteil angebracht werden muss.

Es sind folgende Key-Management-Nachrichten vorgesehen:

- Änderung eines öffentlichen Schlüssels des Benutzers
- Erstmalige Anforderung der Schlüssel des Kreditinstituts
- Erstmalige Übermittlung der Schlüssel des Benutzers
- Schlüsselsperrung durch den Benutzer

Alle Key-Management-Nachrichten müssen eine Initialisierung enthalten und stehen somit am Anfang einer FinTS-Kommunikation. Die durch sie begonnenen Kommunikationen werden stets mit der Kreditinstitutsantwort wieder beendet. Keymanagement-Nachrichten sind damit formal FinTS-Datagramme. Außer dem KeyManagement-Auftrag können in der Nachricht keine weiteren Aufträge versendet werden.

II.6.1.1 Änderung eines öffentlichen Schlüssels des Benutzers

Realisierung Kreditinstitut: verpflichtend
Realisierung Kundenprodukt: verpflichtend

a) Benutzernachricht

♦ **Beschreibung**

Der Änderungsauftrag muss mit dem alten Signierschlüssel signiert werden.

Es muss unterschieden werden, ob die Schlüsseländerung auch das Sicherheitsprofil wechselt oder nicht.

Die folgenden Wechselmöglichkeiten bestehen, falls Sicherheitsprofilwechsel unterstützt sind:

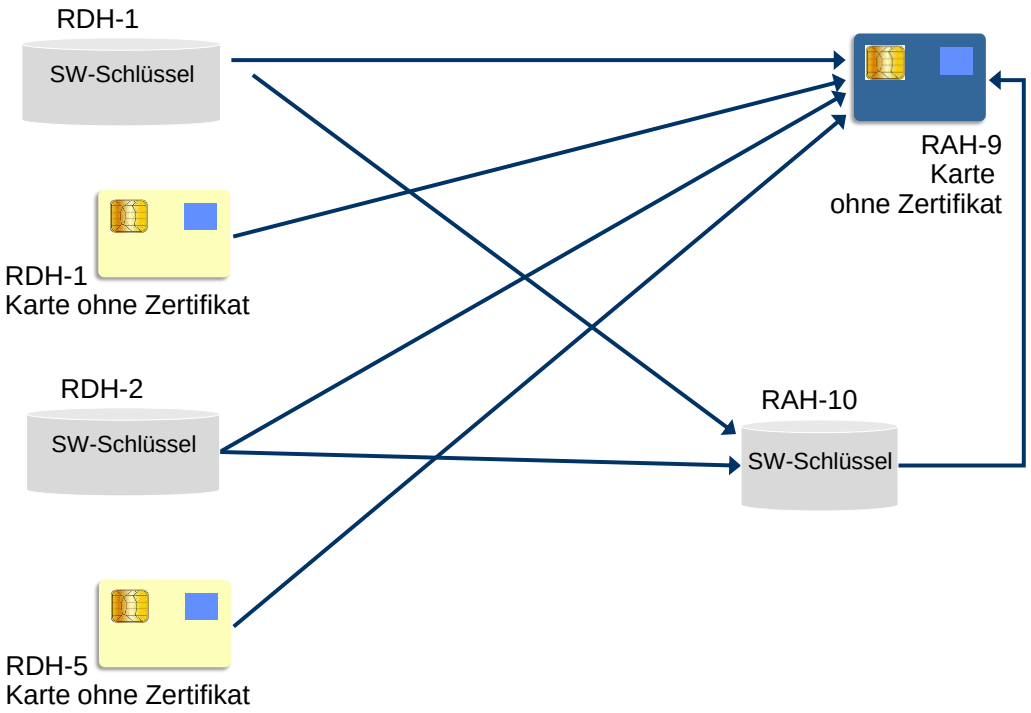


Abbildung 6: Unterstützte Sicherheitsprofilwechsel RDH-1, RDH-2 und RDH-5 auf RAH-9 und RAH-10

Zusammengefasst ergeben sich folgende Wechselmöglichkeiten:

1. ohne Wechsel des Sicherheitsprofils:

   Nach der erfolgreichen Durchführung der Schlüsseländerung wird der vorher aktuelle Schlüssel automatisch gesperrt. Es ist darauf zu achten, dass die Version des neuen Schlüssels höher ist als die des alten Schlüssels.

2. mit Wechsel des Sicherheitsprofils
   (vgl. Abbildung 6):

   Bei einem Sicherheitsprofilwechsel muss der Kunde immer beide HKSAK-Segmente einstellen. Nach der erfolgreichen Durchführung der Schlüsseländerung wird durch das Kreditinstitut mitgeteilt, ob der vorher aktuelle RAH-x bzw. RDH-x-Schlüssel automatisch gesperrt wurde. Diese Nachricht wird mit den RAH-x bzw. RDH-x-Schlüsseln abgesichert. Wurden die RAH-x bzw. RDH-x-Schlüssel institutsseitig nicht gesperrt, wird der Dialog unter Absicherung der RAH-x bzw. RDH-x-Schlüssel beendet. Es ist darauf zu achten, dass die Nummer der RDH-2-Schlüssel 2 ist, die Version kann mit 1 beginnen. Ab RDH-5 und bei RAH-x sind Schlüsselnummer und –version vorgegeben.

Falls das Kreditinstitut nicht in der Lage ist, zwei Schlüsselpaare zu einem Kunden gleichzeitig zu halten und somit die Endenachricht mit den RAH-x bzw. RDH-x-Schlüsseln nicht mehr bedienen kann, ist

dies dem Kundenprodukt durch den Rückmeldungscode 3250 mitzuteilen. Das Kundenprodukt soll dann keine Endenachricht mehr senden und den Bankdatensatz von der RAH-x bzw. RDH-x-Schlüsseldatei löschen.

Es empfiehlt sich, die RAH-x bzw. RDH-x-Schlüssel nach einem erfolgreichen Abschluss des Dialoges durch einen Sperrdialog ungültig zu machen.

Falls der Kunde eine Schlüsseländerungsnachricht sendet, diese aber aus kreditinstitutsinternen Verarbeitungsgründen nicht beantwortet wird, sollte das Kundenprodukt zunächst einen neuen Dialog auf Basis eines der Schlüsselpaare aufbauen. Falls diese Nachricht abgelehnt wird ist ein erneuter Versuch auf Basis eines anderen Schlüsselpaares vorzunehmen. Aus der Reaktion des Kreditinstituts ist für das Kundenprodukt ersichtlich, ob die Schlüsseländerung erfolgreich war oder wiederholt werden muss.
Da es nicht möglich ist, einen DS-Schlüssel, der ja eine natürliche Person identifiziert, über die HBCI-Schlüsseländerung zu ändern, dürften nur "1..2" HKSAK-Segmente eingestellt werden.

Wechsel des Sicherheitsprofils ohne Schlüsselwechsel

Diese Situation tritt bei der Migration von RDH- nach gleichrangigen RAH-Verfahren auf. Beim Übergang von gleichartigen Sicherheitsprofilen (z. B. RDH-9 auf RAH-9 oder RDH-10 auf RAH-10) muss zwar eine erneute Übermittlung der bestehenden öffentlichen Schlüssel durch entsprechende HKSAK-Segmente erfolgen, diese dienen jedoch nur dazu, die Änderung des Sicherheitsprofils bzgl. des Verschlüsselungsalgorithmus (RDH: 2-Key-Triple-DES nach RAH: AES-256) mitzuteilen. Die Schlüsselpaare selbst bleiben unverändert, d. h. weder im Kundenprodukt noch im Kreditinstitut werden Änderungen an den bestehenden Schlüsseln vorgenommen.

Beim Übergang von RDH- auf RAH-Verfahren ergeben sich folgende Möglichkeiten des Schlüsselwechsels (RDH-10 auf RAH-9) bzw. des Wechsel des Verschlüsselungsverfahrens von R**D**H auf R**A**H:

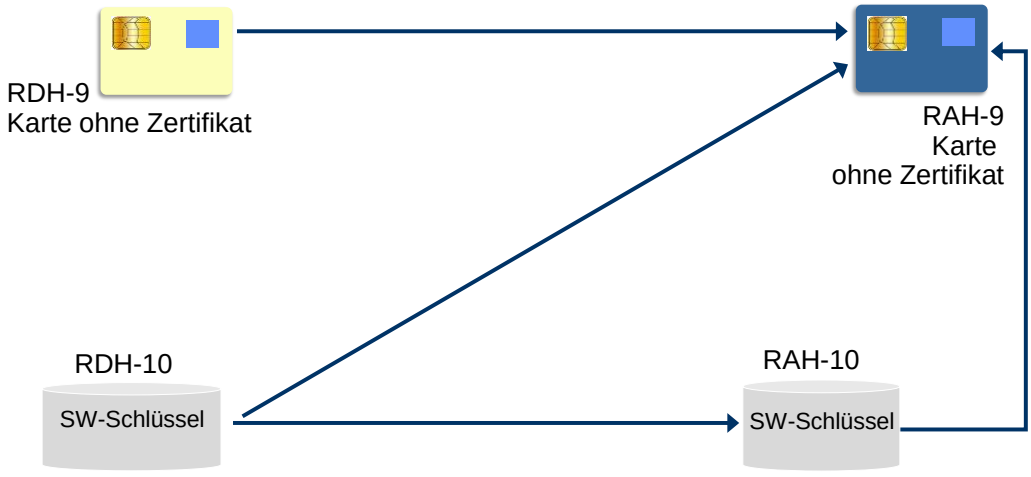


Abbildung 7: Unterstützte Sicherheitsprofilwechsel beim Übergang von RDH- auf RAH-Verfahren

♦ **Belegungsrichtlinien**

Verfahrensbezeichner für das Sicherheitsverfahren RAH

Es ist anzugeben, für welches Sicherheitsverfahren der neue Schlüssel verwendet werden soll.

Schlüsselname

Der Kunde stellt entweder seinen neuen öffentlichen Signierschlüssel, seinen neuen öffentlichen Chiffrierschlüssel oder beide Schlüssel ein.

Modulus, Exponent

Es ist der öffentliche Schlüssel des Benutzers einzustellen.

Zertifikat

Falls für den neuen öffentlichen Schlüssel ein Zertifikat verfügbar ist, kann es dem Kreditinstitut auf diese Weise eingereicht werden, sofern es für das gewählte Sicherheitsverfahren verwendet werden darf.

b) Kreditinstitutsnachricht

♦ **Erläuterungen**

Es werden keine Datensegmente zurückgemeldet. Die Kommunikation wird durch die Kreditinstitutsnachricht explizit beendet.

♦ **Ausgewählte Beispiele für Rückmeldungscodes**

Code	Beispiel
0020	Öffentlicher Schlüssel wurde geändert
3260	Schlüssel weiterhin gültig. Schlüsselsperre wird empfohlen
9010	Schlüsseländerung zur Zeit nicht möglich
9010	Sicherheitsverfahren unterstützt keine öffentlichen Schlüssel
9210	Eingereichter Schlüssel ist mit dem aktuellen Schlüssel identisch

II.6.1.2 Erstmalige Anforderung der Schlüssel des Kreditinstituts

Realisierung Kreditinstitut: optional
Realisierung Kundenprodukt: verpflichtend

Mit Hilfe dieser Nachricht fordert der Benutzer erstmalig die öffentlichen Schlüssel des Kreditinstituts an. Da die Anforderung innerhalb eines eigene Dialoges mit Initialisierung liegt, erhält der Benutzer die aktuellen Bankparameterdaten, die er benötigt, um die unterstützten Verschlüsselungsverfahren des Kreditinstituts in Erfahrung zu bringen. Mit Hilfe dieser Informationen wird der Benutzer in die Lage versetzt, beliebige Nachrichten zu verschlüsseln.

a) Benutzernachricht

♦ **Beschreibung**

Die Nachricht wird anonym gesendet. Sie wird weder signiert noch verschlüsselt.

- **Belegungsrichtlinien**

Verfahrensbezeichner für das Sicherheitsverfahren RAH

Es ist anzugeben, für welches Sicherheitsverfahren der Schlüssel angefordert werden soll.

Schlüsselname

In den Schlüsselnamen ist die Nummer und Version des Schlüssels einzustellen, den das Kundensystem als aktuellen öffentlichen Schlüssel des Kreditinstituts kennt. Falls dieser noch nicht vorliegt, sind beide Felder wegzulassen.

Da bei der Erstinitialisierung noch keine BPD vorliegt, ist es für das Kundensystem evtl. problematisch, zu ermitteln welche Sicherheitsprofile das Kreditinstitut anbietet und - wenn mehrere möglich sind - welches Profil für den Benutzer gilt. Falls dem Benutzer diese Information nicht von seinem Kreditinstitut mitgeteilt wurde, sollte das Kundensystem versuchen, das Sicherheitsmedium zu lesen und daraus das richtige Sicherheitsprofil zu erschließen.

Falls die angegebene Nummer und Version gültig sind, aber nicht mehr aktuelle Schlüssel bezeichnen, sendet das Kreditinstitut diese abgelaufenen Schlüssel, sofern es noch darüber verfügt.

In allen anderen Fällen sendet das Kreditinstitut seine aktuellen Schlüssel. Falls Nummer und Version in der Anforderung abgelaufene Schlüssel bezeichnen und das Kreditinstitut diese Schlüssel nicht senden kann, oder falls Nummer und Version ungültig sind, sollte die Antwortnachricht zusätzlich zu den aktuellen Schlüsseln eine Warnmeldung enthalten.

Typ

Als Schlüsseltyp wird hier immer der Signierschlüssel angegeben.

Generell sollte das Kreditinstitut abgelaufene Bankschlüssel für einen Übergangszeitraum parallel zu den neuen Bankschlüsseln verwenden können. So können auch solche Benutzernachrichten bearbeitet werden, die noch mit dem abgelaufenen Schlüssel verschlüsselt wurden. In diesem Fall sollte das Kreditinstitut dem Benutzer die neuen öffentlichen Schlüssel auch ohne dessen explizite Anforderung innerhalb der Kreditinstitutsantwort zusenden.

b) Kreditinstitutsnachricht

- **Erläuterungen**

  Die Kommunikation wird durch die Kreditinstitutsnachricht explizit beendet.

  In dieser Nachricht sind die öffentlichen Schlüssel des Kreditinstituts explizit als Auftragsantwort enthalten. Außer mit dieser Nachricht können diese Schlüssel vom Kreditinstitut bei Änderungen auch implizit im Rahmen einer normalen Initialisierungsantwort versandt werden.

  Die Nachricht ist nicht verschlüsselt. Falls das Kreditinstitut einen Signierschlüssel besitzt, d. h. seine Nachrichten grundsätzlich signiert, hat es auch diese Nachricht zu signieren, um die Authentizität des Chiffrierschlüssels zu sichern (s. u.).

  Falls das Kreditinstitut seine Nachrichten nicht signiert, erhält der Benutzer nur den öffentlichen Chiffrierschlüssel zurückgemeldet. Auf die Anforderung des Signierschlüssels erhält er einen entsprechenden Rückmeldungscode der Kategorie „Warnungen und Hinweise“, der ihm anzeigt, dass das Kreditinstitut seine Nachrichten nicht signiert. Da die Authentizität des Chiffrierschlüssels nicht gesichert ist, muss diese Nachricht durch einen Ini-Brief an den Benutzer mit dem Hash-Wert des Chiffrierschlüssels begleitet werden (siehe *II.3.1.1.2 Initiale Schlüsselverteilung*).

Falls das Kreditinstitut seine Nachrichten signiert, erhält der Benutzer sowohl den öffentlichen Chiffrier- als auch den Signierschlüssel zurückgemeldet. Die Authentizität des Chiffrierschlüssels ist dabei durch die Signatur gesichert. Die Authentizität des Signierschlüssels ist jedoch nicht gesichert, da das Kundensystem die Echtheit der Signatur nicht prüfen kann. Daher muss in diesem Fall die Nachricht durch einen Ini-Brief mit dem Hash-Wert des Signierschlüssels begleitet werden.

Da die Nachricht unverschlüsselt ist, werden grundsätzlich keine UPD zurück gemeldet.

- **Belegungsrichtlinien**

Verfahrensbezeichner für das Sicherheitsverfahren RAH

Es ist anzugeben, für welches Sicherheitsverfahren der Schlüssel übermittelt wird.

Schlüsselname

Der zurückgemeldete Schlüsselname enthält insbesondere die zugehörige Schlüssel- und Versionsnummer, die das Kundensystem für die Referenzierung des übertragenen neuen öffentlichen Schlüssels verwendet.

Modulus, Exponent

Der neue öffentliche Schlüssel des Kreditinstituts.

Zertifikat

Falls für den neuen öffentlichen Schlüssel ein Zertifikat verfügbar ist und für das gewählte Sicherheitsverfahren verwendet werden darf, wird es in diesem Feld übermittelt.

- **Ausgewählte Beispiele für Rückmeldungscodes**

Code	Beispiel
0020	Auftrag ausgeführt
3310	Kein Schlüssel verfügbar, da Kreditinstitutsnachrichten nicht signiert werden

II.6.1.3 Erstmalige Übermittlung der Schlüssel des Benutzers

Realisierung Kreditinstitut: verpflichtend
Realisierung Kundenprodukt: verpflichtend

Mit Hilfe dieser Nachricht übermittelt der Benutzer erstmalig seine öffentlichen Schlüssel an das Kreditinstitut („Erstinitialisierungsnachricht“).

Da der Absender des öffentlichen Schlüssels den Beweis erbringen muss, dass er auch im Besitz des zugehörigen privaten Schlüssels ist, muss die Nachricht des Benutzers signiert sein.

Das Kreditinstitut darf eine Nachricht nicht ablehnen, nur weil für den Benutzer noch kein öffentlicher Schlüssel in der Schlüsselverwaltung existiert. Falls die normale Signaturprüfung aus diesem Grund negativ verläuft, muss zunächst geprüft werden, ob es sich um eine Erstinitialisierung handelt. In diesem Fall ist der öffentliche Schlüssel aus der Erstinitialisierungsnachricht zu extrahieren und die Signaturprüfung auf der Basis dieses Schlüssels erneut vorzunehmen.

Die Erstinitialisierungsnachricht des Benutzers ist zu verschlüsseln, da die darin enthaltenen benutzerbezogenen Daten (Kunden-ID, Benutzerkennung) als vertraulich einzustufen sind. Dies erfordert, dass sich der öffentliche Chiffrierschlüssel des Kreditinstituts schon vor dem Senden der Erstinitialisierung im Besitz des Benutzers befinden muss. Ferner muss dem Benutzer das Verschlüsselungsverfahren bekannt sein, das ihm in den Bankparameterdaten mitgeteilt wird. Um dem Benutzer diese Daten vorab zukommen zu lassen, bieten sich folgende Lösungen an:

- Das Kreditinstitut sendet dem Benutzer einen Datenträger (z. B. Diskette oder USB-Stick) zu, der die Schlüssel und die aktuelle BPD enthält, wie in *II.3.1.1.2 Initiale Schlüsselverteilung* beschrieben.
- Der Benutzer sendet die Key-Management-Nachricht „Erstmalige Anforderung der Schlüssel des Kreditinstituts“ (siehe *II.6.1.2 Erstmalige Anforderung der Schlüssel des Kreditinstituts*). Diese Nachricht wird begleitet von einem Ini-Brief.

Um die wiederholte Ausführung unberechtigter Initialisierungsversuche zu verhindern, sind kreditinstitutsseitig folgende Vorkehrungen zu treffen:

- Die Benutzerkennung sollte bei Verwendung des RAH-Verfahrens nicht durch benutzerindividuelle Merkmale (z. B. Kontonummer) hergeleitet werden können.
- Eine erneute Erstinitialisierung ist nur zulässig, wenn zuvor eine Sperrung der Schlüssel des Benutzers erfolgt ist. In allen anderen Fällen ist eine erneute Erstinitialisierungsnachricht abzulehnen.

Auf der Chipkarte können Kommunikationszugänge abgelegt werden (siehe IV). Da pro Kreditinstitut jedoch mehrere Kommunikationszugänge gespeichert sein können (z. B. HTTP und SMTP), muss ein Kundensystem zunächst prüfen, ob für dieses Kreditinstitut bereits die Schlüssel eingereicht wurden, bevor eine erstmalige Übermittlung der Schlüssel des Benutzers durchgeführt wird. Für den Fall, dass das Kundensystem die Schlüssel dennoch sendet, sollte das Kreditinstitut die Warnung 3330 „Schlüssel liegen bereits vor“ zurückmelden.

a) Benutzernachricht

- **Beschreibung**

  Die Nachricht muss signiert und verschlüsselt werden.

  Der Benutzer stellt seine öffentlichen Schlüssel ein. Dies können Signier-, Chiffrier- oder Authentifikationsschlüssel sein.

  Die Authentizität des Chiffrierschlüssels ist dabei durch die Signatur gesichert. Die Authentizität des Signierschlüssels ist jedoch nicht gesichert, da das Kreditinstitut die Echtheit der Signatur nicht prüfen kann. Daher muss die Nachricht durch einen Ini-Brief an das Kreditinstitut mit dem Hash-Wert des Signierschlüssels begleitet werden (siehe *II.3.1.1.2 Initiale Schlüsselverteilung*).

- **Belegungsrichtlinien**

  Sicherheitsverfahren

  Es ist anzugeben, für welches Sicherheitsverfahren der Schlüssel übermittelt wird.

  Schlüsselname

  Der zurückgemeldete Schlüsselname enthält insbesondere die zugehörige Schlüssel- und Versionsnummer, die das Kundensystem für die Referenzierung des übertragenen neuen öffentlichen Schlüssels verwendet.

  Modulus, Exponent

  Diese Datenelementgruppe enthält den neuen öffentlichen Schlüssel des Kreditinstitutes.

  Zertifikat

  Falls für den neuen öffentlichen Schlüssel ein Zertifikat verfügbar ist und für das gewählte Sicherheitsverfahren verwendet werden darf, wird es in diesem Feld übermittelt.

b) Kreditinstitutsnachricht

Die Nachricht ist bei erfolgreicher Ausführung signiert. Sie ist stets unverschlüsselt, da der Chiffrierschlüssel des Benutzers erst nach erfolgreicher Verifikation des Ini-Briefs gültig und damit verwendbar ist.

Da die Nachricht unverschlüsselt ist, werden grundsätzlich keine UPD zurück gemeldet.

♦ **Beschreibung**

Die Ablehnung der Erstinitialisierungsnachricht darf aus sicherheitstechnischen Aspekten im Rahmen der Rückmeldungscodes nicht inhaltlich begründet werden. Fehlermeldungen, die sich auf den syntaktischen Aufbau der Nachricht bzw. der Segmente beziehen, sind hiervon unberührt.

♦ **Erläuterungen**

Es werden keine Datensegmente zurückgemeldet. Die Kommunikation wird durch die Kreditinstitutsnachricht explizit beendet.

♦ **Ausgewählte Beispiele für Rückmeldungscodes**

Code	Beispiel
0010	Öffentlicher Schlüssel wurde entgegengenommen
0020	Öffentlicher Schlüssel wurde freigeschaltet
0020	Benutzer wurde freigeschaltet
3330	Schlüssel liegen bereits vor
9010	Auftrag abgelehnt

II.6.1.4 Schlüsselsperrung durch den Benutzer

Realisierung Kreditinstitut: verpflichtend
Realisierung Kundenprodukt: verpflichtend

Diese Nachricht beschreibt die Anforderung zum Sperren der Schlüssel durch den Benutzer und die Bestätigung der Schlüsselsperrung durch das Kreditinstitut (vgl. *II.3.2 Schlüsselsperrung*).

a) Benutzernachricht

♦ **Beschreibung**

Es werden immer alle Schlüssel gesperrt. Eine selektive Schlüsselsperrung (z. B. nur Chiffrierschlüssel) ist gegenwärtig nicht zulässig.

Die Nachricht muss signiert sein. Nicht-signierte (anonyme) Schlüsselsperrungen sind nicht vorgesehen.

Bei Verlust des Sicherheitsmediums liegen dem Benutzer u. U. die zur Durchführung der Sperrung erforderlichen Daten (Schlüsselnummer und -version) nicht vor. In diesem Fall ist die Sperre über einen anderen Weg (z. B. Callcenter) durchzuführen.

Beim RAH-Verfahren muss der Benutzer nach einer Schlüsselsperrung zur Entsperrung eine erneute Erstinitialisierung durchführen.

♦ **Belegungsrichtlinien**

Verfahrensbezeichner für das Sicherheitsverfahren RAH

Es ist anzugeben, für welches Sicherheitsverfahren die Schlüssel gesperrt werden sollen.

Benutzerschlüsselname

Es sind die Identifikationsmerkmale des zu sperrenden Signierschlüssels einzustellen, unabhängig davon, dass grundsätzlich immer sowohl der/die Signier- als auch der Chiffrierschlüssel gesperrt werden (siehe *II.3.2 Schlüsselsperrung*).

Sperrgrund

Es ist der Grund für die Sperre anzugeben. Dies kann z. B. die Kompromitierung der Benutzerschlüssel oder der Verdacht darauf sein.

Annulierungszeitpunkt

Enthält optional Datum und Uhrzeit, ab welcher der Schlüssel nicht mehr gültig ist.

Es ist zu beachten, dass eine terminierte Sperre nicht von allen Kreditinstituten unterstützt wird. Das Kundensystem sollte den Benutzer auf diesen Sachverhalt hinweisen.

b) Kreditinstitutsnachricht

- **Erläuterungen**

Beim RAH-Verfahren wird im Anschluss an die Sperrnachricht die Antwortnachricht des Kreditinstituts nicht chiffriert, aber signiert (sofern das Kreditinstitut grundsätzlich signiert).

Diese Verfahren gelten nur bei einer erfolgreichen Sperrung. Bei einer fehlgeschlagenen Sperrung ist die Kommunikation gesichert zu Ende zu führen, da die Schlüssel des Benutzers weiterhin aktiv sind.

- **Ausgewählte Beispiele für Rückmeldungscodes**

Code	Beispiel
0020	Schlüssel wurde erfolgreich gesperrt
9010	Schlüssel ist bereits gesperrt
9010	Terminierte Sperren werden nicht unterstützt
9210	Unbekanntes Sperrenkennzeichen
9210	Sperrdatum liegt zu weit in der Zukunft

III. SECODERINTEGRATION

III.1 Einleitung

In diesem Abschnitt wird die Integration von Chipkartenlesern mit Secoder-Funktion in das FinTS V4.1 Protokoll beschrieben. Beim Secoder werden zusätzlich sicherheitsrelevante Informationen wie z. B. Visualisierungsdaten oder eine Visualisierungsbestätigungssignatur übertragen. Diese finden bei FinTS V4.1 in einem eigenen BPD-Abschnitt zum Sicherheitsverfahren „Secoder“ Platz. In der aktuellen Ausbaustufe wird im Secoder nur die Anwendung „aut“ zur Bildung von Signaturen unterstützt.

Das Secoder-Sicherheitsverfahren verhält sich vom Kommunikationsablauf her wie eine normale FinTS-Transaktion im Ein-Schritt-Verfahren. Informationen bzgl. Nachrichtenaufbau und Dialogablauf sind dem Dokument [Formals] zu entnehmen.

Ob ein Kreditinstitut Secoder-Sicherheitsverfahren anbietet, erkennt das Kundenprodukt in den Bankparameterdaten am Vorhandensein der jeweiligen Strukturen zum Secoder (vgl. Kapitel III.2.4 „Parameterdaten Secodersignatur“)

Grundsätzlich können mit Secoder-Sicherheitsverfahren alle im Dokument [Messages] aufgeführten Geschäftsvorfälle verwendet werden. Dies gilt auch für verbandsindividuelle Erweiterungen. Welche Geschäftsvorfälle konkret mit welchem Secoder-Sicherheitsverfahren zulässig sind, teilt das Kreditinstitut in den Bankparameterdaten in der Liste der erlaubten Geschäftsvorfälle mit.

Als Verschlüsselungsverfahren wird bei dem derzeit einzigen Secoder-Sicherheitsverfahren=811 die HBCI-Verschlüsselung verwendet.

Bei Secoder-Sicherheitsverfahren handelt es sich wie bei HBCI-Sicherheit um Ein-Schritt-Verfahren, d. h. der Geschäftsvorfall wird in einem Prozess-Schritt zusammen mit allen benötigten Signaturinformationen eingereicht und somit in einem Dialogschritt bestehend aus Auftrag und Antwort wie ein Geschäftsvorfall komplett abgewickelt.

Durch die neu eingeführten Secoder Metadaten (vgl. [Master], Abschnitt „Definitionen“), die über die BPD-Parameter zum Secoder abgebildet werden, können die Secoder-Sicherheitsverfahren für fortgeschrittene Signaturen ebenfalls das Ein-Schritt-Verfahren einsetzen.

III.1.1 Secodervisualisierung

Bereits durch die Integration von Zwei-Schritt-TAN-Verfahren wie dem chipTAN- oder mobileTAN-Verfahren wurden Teile des Auftrags als Challenge visualisiert und durch den Kunden bestätigt. Bei Secoder-Sicherheitsverfahren wird dieser Ansatz weiter verfeinert, da hier ggf. größere Datenmengen zu visualisieren sind. Beim Secoder können bei maximaler Puffergröße und einer 2 x 16 Anzeigeeinheit z. B. bis zu 9 Elemente im Display durchgeblättert und bestätigt werden (Secodervisualisierung).

Bei Einzelaufträgen ist die Secodervisualisierung auf einfache Art zu bewerkstelligen, wobei die Bankanwendung die Secodertexte frei definieren kann. So können beispielsweise auch in Anlehnung an die Belegungsrichtlinien von HHD V1.4 Daten aus dem Auftrag in geeigneter Weise angezeigt und bestätigt werden. Der Secoder erlaubt mit seinen Techniken hierfür noch weitreichendere Möglichkeiten als das HHD.

Bei Sammelaufträgen müssen Konstrukte geschaffen werden, deren Secodervisualisierung und Bestätigung dem Kunden einerseits noch zumutbar sind und ein vertretbares Sicherheitsniveau garantieren. Solche Lösungen können jedoch immer nur einen Kompromiss darstellen.

Als Möglichkeiten der Auswahl der Secodervisualisierungsdaten bei solch großen Datenmengen kommen im Wesentlichen Summenwerte über verschiedene Daten wie Anzahl der Sätze oder Beträge in Frage. Auch die Signatur von Hashwerten und Vergleich mit einem zur Verfügung gestellten „Hashwert-Tool“ ist Praxis. Die Thematik ist nicht neu und im Firmenkundengeschäft generell zu betrachten.

Die gesamte Themenstellung ist jedoch nicht Inhalt dieser Spezifikation, sondern Betreiber-spezifisch zu lösen. Die Möglichkeiten hierfür sind durch die Verwendung der Metadaten in der BPD gegeben.

III.1.2 Secoder-Integration in FinTS

Zur Unterstützung des Secoders werden neue Funktionalitäten in das FinTS-Protokoll integriert wie z. B. die Visualisierung eines Teils der Transaktionsdaten und das daraus resultierende Secoder-Kryptogramm. Die Secoder-Integration hat aber auch inhaltliche Auswirkungen, wie in den folgenden Abschnitten dargestellt ist.

III.1.2.1 Secoder-Applikationen und Verwendungsmöglichkeiten

In FinTS wird derzeit nur die folgende Secoder-Applikation unterstützt:

- „aut“ Signaturapplikation unter Verwendung des AUT-Schlüssels

Die Applikation „aut“ wird im Element „Variante des Secoderverfahrens“ durch den Wert 811 gekennzeichnet.

In den folgenden Prozessabläufen wird davon ausgegangen, dass bei der Signaturbildung sowohl die Auftragsdaten (in Form eines übergebenen Hashwerts über die Auftragsdaten) kombiniert mit den Secoder-Visualisierungsdaten signiert werden als auch eine Visualisierungsauthentikationssignatur gebildet wird, durch das FinTS-Protokoll also zwei Signaturen zu übertragen sind. Dies wird durch zwei Secoder-Aufrufe der Applikationen aut/aut erreicht.

III.1.2.2 Secoder-Verwaltung / Abfrage der Secoder-Eigenschaften

Grundsätzlich muss ein Kreditinstitut die physischen Eigenschaften des vom Kunden genutzten Secoders nicht kennen. Durch die Einführung einer Metasprache (siehe nächster Abschnitt) erfolgt eine logische Entkopplung der fachlichen von den physischen Eigenschaften. Durch die Angabe „811“ im Element „Variante des Secoderverfahrens“ gibt das Kreditinstitut zwar die unterstützten Varianten von Secoder-Sicherheitsverfahren vor, es werden dort aber keine physischen Eigenschaften des Secoders vorgegeben. Die Secoder-Anwendungsfunktion ist dafür verantwortlich, aus den übergebenen Metadaten passende SecCmds für den Secoder-Aufruf zu erzeugen. Da mit der aktuellen Version der Secoder-Spezifikation 2.2 (Erratum vom 16.08.2012) auch größere Displays als 2 x 16 unterstützt werden, muss die Secoder-Anwendungsfunktion entweder die physischen Eigenschaften des angeschlossenen Secoders kennen (per SecCmd „SECODER INFO“) oder nur jeweils 2 x 16 Zeichen ausgeben.

III.1.2.2.1 Einführen einer Metasprache

Da es aus vielerlei Gründen nicht erwünscht ist, dass Online-Banking-Applikationen direkt Secoder-spezifische Kommandos aufbauen und übertragen, bietet sich die Schaffung einer Metasprache (vgl. Definiton „Secoder MetaData“ in [Master], Abschnitt “Definitionen“) an, die auf hohem Abstraktionsgrad all die Möglichkeiten abbildet, die ein Secoder darstellen kann. Hierzu gehört beispielsweise auch die Information, ob ein Element nur bestätigt oder vom Kunden eingetippt werden soll. Die MetaData entsprechen daher inhaltlich in etwa den Datensätzen DSx im Input von DATA CONFIRMATION.

Die Metasprache kapselt auch die beiden Secoder-Anwendungsaufrufe aut/aut zur Signaturbildung. Bei FinTS wird die Metasprache mit den Sprachmöglichkeiten der Bankparameterdaten abgebildet.

Die von der Online-Banking-Applikation erzeugten MetaData müssen von einer geeigneten Secoder-Anwendungsfunktion in die Secoder-spezifischen Kommandos umgesetzt werden. Dies kann zum einen ein Bestandteil eines FinTS-Kundenproduktes sein, im Bereich Internet-Banking z. B. aber auch eine Webserver-Applikation mit einem PlugIn oder Applet auf dem Kunden-PC (für die Ansteuerung des Secoders wird in jedem Fall eine aktive Komponente auf dem Kunden-PC benötigt).

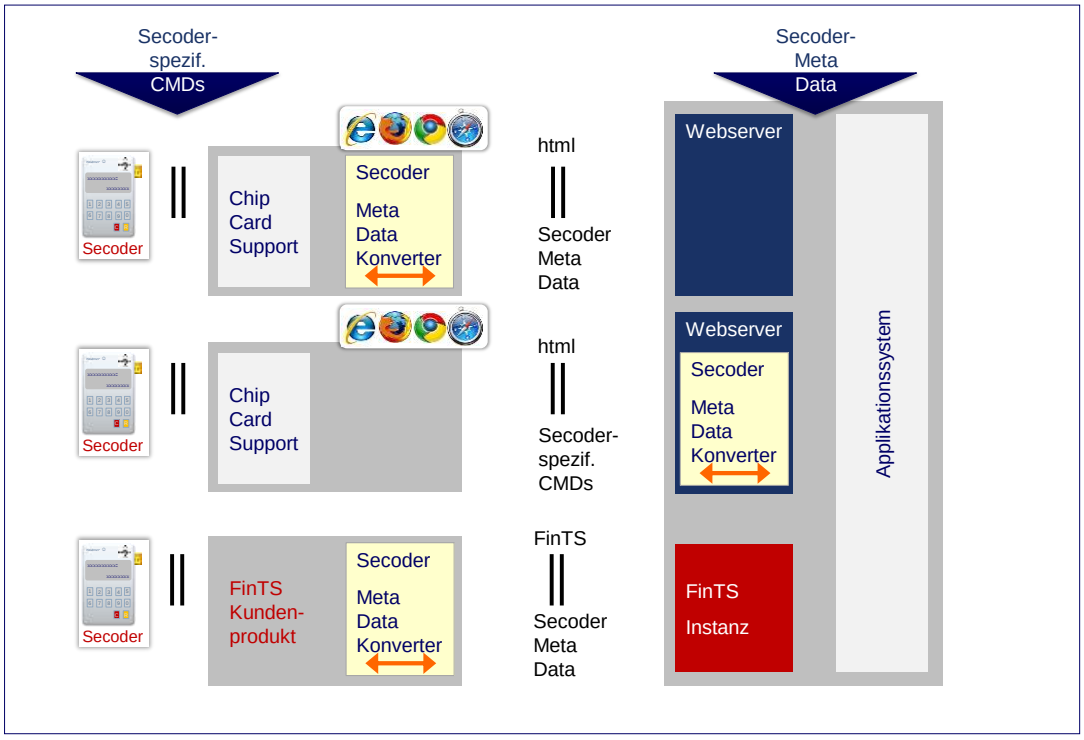


Abbildung 8: Mögliche logische Architekturen zur Integration des Secoders über eine Metadatenschnittstelle

III.2 Verfahrensbeschreibung

III.2.1 Allgemeines

Es gelten die in [Formals] und im übertragenen Sinn die in [Belegung] aufgeführten Formate und Belegungsrichtlinien.

Für den Einsatz von Secoder-Sicherheitsverfahren gelten zusätzlich die folgenden allgemeinen Festlegungen:

- Zur eindeutigen Bezeichnung der Variante des Secoder-Verfahrens wird der Nummernkreis 800 bis 899 zur Kodierung verwendet. Auch die Verknüpfung von Code und Verfahren ist Teil dieser Spezifikation und wird in der BPD festgelegt.
- Mit dem Rückmeldungscode 3921 und Rückmeldungsparametern (vgl. [RMCode]) werden dem Kunden in der Initialisierungsantwort die für ihn zugelassenen Secoder-Sicherheitsverfahren mitgeteilt. Als Bezugssegment für das Rückmeldungssegment wird das Segment *ProcPreparation* verwendet.

  Der Kunde übermittelt im Signaturkopf der Initialisierungsnachricht, mit welchem konkreten Secoder-Sicherheitsverfahren er den Dialog führen will. Das konkrete Secoder-Sicherheitsverfahren darf während des Dialogs nicht gewechselt werden (Näheres hierzu siehe Abschnitt III.3).
- Beim Einsatz von Mehrfach-Signaturen gilt ein konkretes Secoder-Sicherheitsverfahren für den gesamten Dialog des jeweiligen Benutzers. Jeder Benutzer kann ein eigenes konkretes Secoder-Sicherheitsverfahren verwenden, dieses darf im Kontext einer Mehrfach-Signatur-Einreichung jedoch nicht gewechselt werden.
  Im Falle eines nicht zugelassenen Wechsels des Secoder-Sicherheitsverfahrens muss das Kreditinstitut den Dialog mit Rückmeldungscode 9957 „Wechsel des Secoder-Sicherheitsverfahrens bei Mehrfach-Signaturen nicht erlaubt“ beenden.
- Die Signierung von Kreditinstitutsnachrichten wird bei Secoder-Sicherheitsverfahren momentan nicht unterstützt.

III.2.2 Secoder-Sicherheitsverfahren zur Secoder-Integration ab FinTS 4.1

Im Folgenden wird ausschließlich das derzeit spezifizierte Secoder-Sicherheitsverfahren 811 betrachtet. Secoder-Kryptogramme (Definition vgl. [Master], Abschnitt „Definitionen“) ersetzen die ansonsten in HBCI verwendeten RAH-Signaturen, d. h. der Hashwert über die zu signierenden Auftragsdaten wird in der FinTS-Instanz gebildet, an den Secoder übertragen, mit Visualisierungsdaten angereichert und dort signiert. In einem zweiten Aufruf des Secoders werden zur *Visualisation Authentication* die Visualisierungsdaten zusammen mit einem definierten Visualisierungsbestätigungssignaturfüllwert (Konstante) signiert. Der Visualisierungsbestätigungssignaturfüllwert bewirkt, dass institutsseitig eindeutig festgestellt werden kann, dass der Secoder sich zum Zeitpunkt der Signaturbildung im Applikationsmodus befand, falls ein Secoder verwendet wurde (vgl. [Secoder]). Ein konkretes Secoder-Sicherheitsverfahren wie z. B. 811 bezeichnet in diesem Sinne also ein konkretes durch den Secoder durchzuführendes und dort definiertes, ein-schrittiges Signaturverfahren.

Die Arbeitsweise des Secoders sieht zur Signaturbildung vor, dass zusammen mit dem Hashwert über die Auftragsdaten eine Auswahl von Transaktionsdaten in Form von Secoder-spezifischen Kommandos (SecCmds) z. B. über die USB-Schnittstelle an den Secoder gesendet werden. Befindet sich der Secoder im Applikationsmodus – d. h. ist z. B. beim Secoder-Sicherheitsverfahren 811 die Signatur-Applikation „aut“ selektiert – so werden die relevanten Daten im Secoder-Display angezeigt. Bestätigt der Kunde diese Daten mit Hilfe der Secoder-Tastatur, so gehen diese zusammen mit anderen Informationen in die Bildung der Signatur mit ein. Die Signatur selbst wird als Antwort an die aufrufende Secoder-Anwendungsfunktion im PC zurückgegeben und kann dort in das FinTS-Protokoll im Element „Visualisierungsbestätigungssignaturdaten“ (Tagname *VisualizationSignatureData*) integriert werden.

III.2.2.1 Secoder-Sicherheitsverfahren

Beim Ein-Schritt-Verfahren werden Daten und sämtliche Sicherheitselemente in einem Dialogschritt an das Kreditinstitut gesendet und von dort beantwortet. Alle Informationen zur Absicherung des Auftrags wie z. B. die am Secoder zu signierenden Auftragselemente sind lokal in der FinTS-Instanz über die BPD bekannt; es wird also kein zwei-schrittiges Challenge-Response-Verfahren wie bei chipTAN oder mobile TAN (vgl. [PINTAN]) benötigt.

Mit Secoder-Sicherheitsverfahren werden keine Verfahren konkret spezifiziert – es erfolgt nur eine abstrakte Definition des Ablaufs, der über Parameter gesteuert wird. Der Ablauf selbst ist für alle Secoder-Sicherheitsverfahren identisch. Die Parametrisierung eines konkreten Secoder-Sicherheitsverfahrens erfolgt über die Bankparameterdaten zum Secoder-Sicherheitsverfahren.

Bei Verwendung von Mehrfach-Signaturen wird innerhalb eines Ablaufs das Secoder-Sicherheitsverfahren durch den Dialogführer der ersten (und ggf. einzigen) Kommunikation für alle beteiligten Benutzer festgelegt.

Durch Verwendung der Bankparameterstruktur *Parameter Secodersignatur* ist die abstrakte Beschreibung aller verfügbaren konkreten Secoder-Sicherheitsverfahren in der BPD möglich, die über das Element *Variante des Secoderverfahrens* referenziert werden (Details siehe Kapitel III.2.4 „*Parameterdaten Secodersignatur*“). Bei der Verwendung von Mehrfach-Signaturen kann jeder beteiligte Benutzer ein eigenes konkretes Secoder-Sicherheitsverfahren verwenden – die Verfahren können also innerhalb einer Nachricht unterschiedlich sein[1].

Secoder-Sicherheitsverfahren 811:

Bei dieser Variante wird der Auftrag inklusive aller Sicherheitsinformationen in einem Kommunikationsschritt an das Kreditinstitut übertragen. Abhängig vom Verwendungszweck existieren folgende Signaturvarianten:

[1] Da es in der aktuellen Kommunikation nur einen Boten geben kann, müssen die zulässigen konkreten Secoder-Sicherheitsverfahren der weiteren Benutzer bereits vorab über separate Kommunikationen (und entsprechende Rückmeldungscodes 3921) festgelegt worden sein.

- Secoder-Signatur:
  Sie dient der Einreichung von Auftrag und Secoder-Sicherheitsinformationen (VisDataSig und VisAuthSig, vgl. [Master], Abschnitt „Definitionen“) und wird durch das Kreditinstitut beantwortet. Kennzeichen der Secoder-Signatur ist also, dass die Signaturbildung inklusive Secoder-Visualisierung geschieht. Secoder-Signaturen werden sowohl im Rahmen der Initialisierung (mit Visualisierung) als auch bei der Auftragsverarbeitung benutzt.
- HBCI-RAH-Signatur:
  HBCI-Signaturen werden eingesetzt, wenn es sich um keine Secoder-Signatur, sondern um eine RAH-Signatur ohne Secoder-Visualisierung handelt.

  Dies ist z. B. bei der Initialisierung ohne Visualisierungsdaten und bei Key-Management-Geschäftsvorfällen wie der Schlüsseleinreichung und –Änderung der Fall, bei denen keine Secoder-Visualisierung erfolgt (vgl. Abschnitt III.3.1.3 „Key-Management bei Secoder-Sicherheitsverfahren“).

  HBCI-Signaturen können auch bei einer späteren Verwendung von Institutssignaturen genutzt werden, da Instituts-seitig keine (Secoder-)Visualisierung verwendet wird.

III.2.2.2 Abläufe für Secoder-Sicherheitsverfahren

Die Abläufe zur Abwicklung der unterschiedlichen Secoder-Sicherheitsverfahren unterscheiden sich je nach gewählter Variante. Konkret ist in der vorliegenden Version nur der folgende Ablauf unterstützt:

	Variante	**Sicherheitsverfahren mit Secoder:**
Ablauf 1:	811	Nutzung der Secoder-Applikation „aut“ mit Ein-Schritt-Secoder-Sicherheitsverfahren ohne Institutssignatur

Die folgenden Abläufe sind bezogen auf die einzelnen Prozessschritte exakt in der beschriebenen Form umzusetzen; die Bildung von anderen Derivaten ist nicht zugelassen.

Bei allen Abläufen wird davon ausgegangen, dass sich nur ein signaturpflichtiger FinTS-Auftrag in der Nachricht befindet. Dabei kann es sich auch um einen Sammelauftrag handeln.

Innerhalb einer Kommunikation ist es grundsätzlich möglich aber nicht verpflichtend, dass mehrere in sich abgeschlossene Abläufe hintereinander durchgeführt werden.

Es gelten hierbei als Rahmenbedingungen die für die gesamte Kommunikation getroffenen Festlegungen, z. B. dass das Secoder-Sicherheitsverfahren innerhalb einer Kommunikation nicht gewechselt werden darf.

Bei der Verwendung von Mehrfach-Signaturen sind Aufträge, bei denen mindestens eine Secoder-Signatur fehlerhaft ist, Kreditinstituts-seitig zu verwerfen.

III.2.2.2.1 Ablauf für Variante des Secoderverfahrens = 811

Bei diesem Szenario wird von einem ein-schrittigen Verfahren ausgegangen, d. h. die im Secoder zu visualisierenden Daten müssen dem Kundensystem zuvor über die BPD mitgeteilt werden.

Es wird davon ausgegangen, dass der Benutzer vor der ersten Verwendung dieses Secoder-Sicherheitsverfahrens vollständig initialisiert ist, d. h. es muss über das Standard RAH-Verfahren ein Schlüsselaustausch erfolgt sein (vgl. hierzu auch Abschnitt III.3.1.3 „Key-Management bei Secoder-Sicherheitsverfahren“).

Die Kreditinstitutsantwort enthält beim Secoder-Sicherheitsverfahren 811 keine Institutssignatur.

a) Initialisierung

Bei der Initialisierung werden optional Daten im Secoder visualisiert und es erfolgt dann eine fortgeschrittene Signatur über die Anmeldedaten und ggf. Visualisierungsdaten mit der Secoder-Anwendung „aut“.

Im folgenden Prozess-Beispiel wird von diesem Fall ausgegangen, d. h. im Rahmen der Initialisierung sollen im Secoder Daten angezeigt und bestätigt werden.

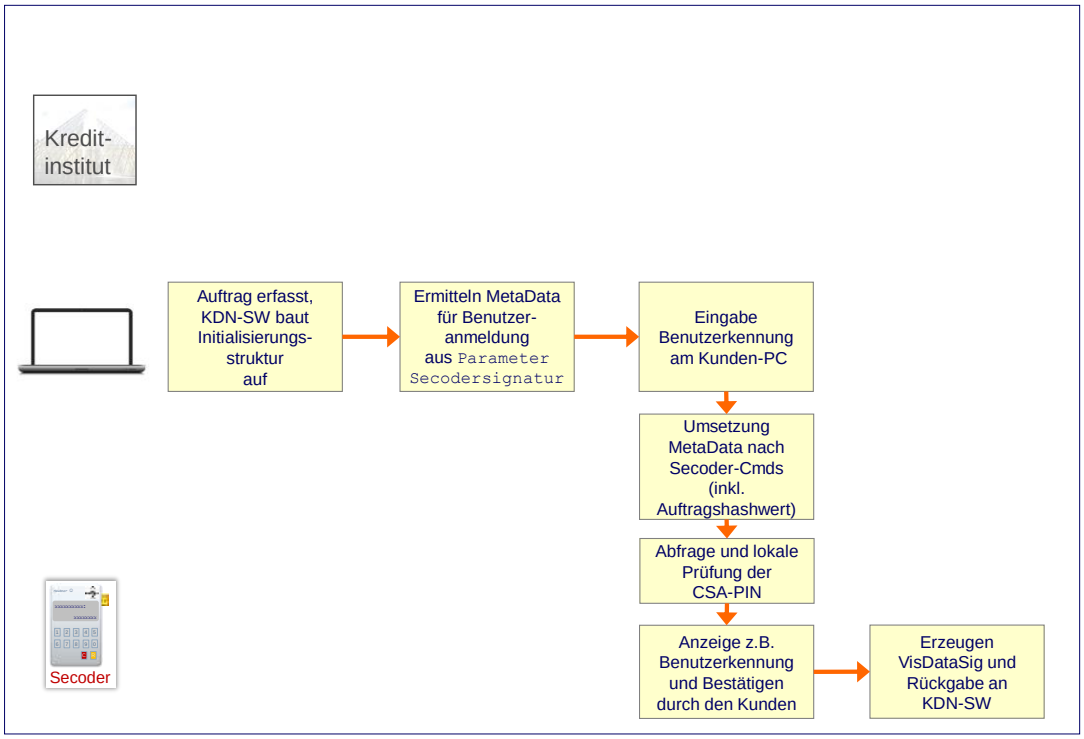


Abbildung 9: Initialisierung beim Secoder-Sicherheitsverfahren 811 (1 von 2)

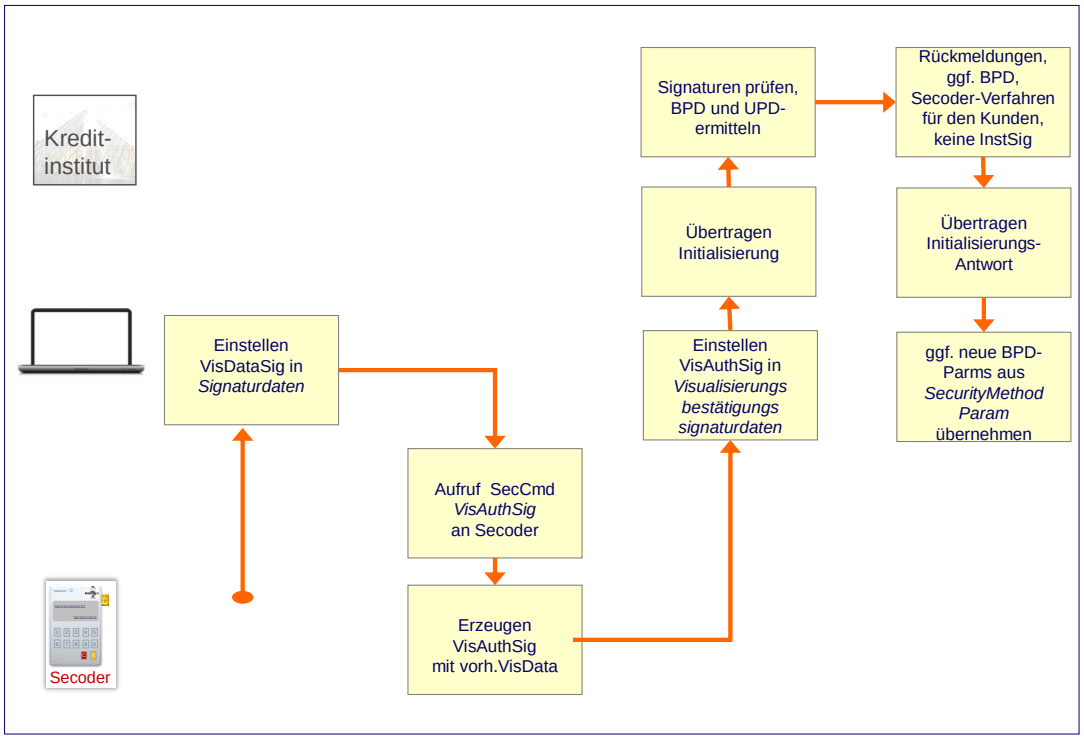


Abbildung 10: Initialisierung beim Secoder-Sicherheitsverfahren 811 (2 von 2)

Der vollständige Ablauf sieht folgendermaßen aus:

Initialisierung beim Secoder-Sicherheitsverfahren 811

Ausgangszustand:

- Der Kunde ist vollständig initialisiert und seine Karte ist frei geschaltet (über Zertifikate oder Ini-Brief-Austausch). Dieser Vorgang wird über das Standard RAH-Verfahren durchgeführt.
- Der Benutzer hat in der Initialisierungsnachricht durch entsprechende Belegung des Elementes *Secoder-Signatur* das Secoder-Sicherheitsverfahren 811 für die gesamte Kommunikation gewählt.
- Im Kundensystem ist eine aktuelle BPD gespeichert, aus der die Steuerungsinformationen zum Aufbau der MetaData ermittelt werden können. Ggf. muss die FinTS-Instanz die aktuelle BPD über einen anonymen Dialog ermitteln.

Schritt 1a Init-Request	→	Benutzerauthentikationsdaten senden Das Kundensystem baut eine standardmäßige Initialisierungsnachricht auf. Als Signaturalgorithmus wird ein geeignetes (BPD) Secoder-Sicherheitsverfahren selektiert und in die Secoder-spezifischen Parameter eingestellt. Über zu signierende Elemente wird entsprechend den Angaben im Sicherheitsprofil der Auftragshashwert erzeugt. Falls die Länge des Auftragshashwerts nicht der Blocklänge entspricht, wird dieser wird mit Nullen aufgefüllt. Es erfolgt ein Zugriff auf den Secoder im Default-Modus. Beim ersten Zugriff wird der Kunde aufgefordert, sein CSA-Passwort einzugeben.

		Der PIN Sicherheitszustand wird über den gesamten Ablauf hinweg gehalten, außer er wird durch eine Benutzeraktion unterbrochen. Auch bei dem Wechsel zwischen Transparent- und Applikationsmodus im Secoder muss das CSA-Passwort nicht neu eingegeben werden. Anschließend wird der aktuelle Signaturzähler des Signier-Schlüssels ermittelt, inkrementiert und in das Element Signatur-ID in den Secoder-spezifischen Parametern eingestellt. Auch die restlichen benötigten Werte wie z. B. CID, Schlüsselnummer, Schlüsselversion und ggf. Kunden-Zertifikat des Signierschlüssels werden auf diese Weise von der Karte gelesen und für die spätere Auftragsverarbeitung im Kundensystem gespeichert. Auf Basis der Informationen aus den *Parameterdaten Secodersignatur* erzeugt das Kundensystem die benötigten SecCmds, um Visualisierungsinformationen und den Auftragshashwert zu übergeben und am Secoder im Applikationsmodus „aut“ eine VisData-Signatur mit Visualisierung der relevanten Daten durchzuführen. Hierbei wird der VisData-Puffer sukzessive mit den Ergebnissen des Visualisierungsprozesses aufgefüllt. Nach der letzten Bestätigung des Benutzers wird – beginnend mit dem Auftragshashwert, der als Initialwert zwischengespeichert wurde – in der Karte ein Hashwert über den VisData-Puffer gebildet. Das Ergebnis dieser Hashoperation wird an den Secoder zurückgegeben und ersetzt den Inhalt des VisData-Puffers. Über den erzeugten Hashwert wird dann eine VisData-Signatur mit dem Signierschlüssel gebildet und vom Secoder an die aufrufende Applikation zurückgegeben. Die Signatur wird dort in die bereits vorbereitete Struktur *Secoder-Signatur* eingestellt. Das Kundenprodukt ruft den Secoder nochmals im Applikationsmodus „aut“ zur Visualisierungsbestätigung auf (*VisAuthSig*) und übergibt hierzu den konstanten Visualisierungsbestätigungssignaturfüllwert "SECODERSECODERSECODE". Der noch im VisData-Puffer vorhandene Hashwert über die Secoder-Visualisierungsdaten wird an den ersten Stellen mit dem Visualisierungsbestätigungssignaturfüllwert überschrieben und über dieses Resultat eine VisAuth-Signatur gebildet. Die erzeugte VisAuth-Signatur wird an das Kundenprodukt zurückgegeben und dort in das Datenelement *Visualisierungsbestätigungssignaturdaten* eingestellt. Die gesamte Initialisierungsnachricht wird HBCI-verschlüsselt zum Institut übertragen.
Schritt 1b Init-Response	←	Authentikationsantwort senden Nach Überprüfung von *VisDataSig* und *VisAuthSig* wird eine Initialisierungsantwort aufgebaut. Da beim Secoder-Sicherheitsverfahren 811 keine Institutssignatur verwendet wird, wird die Nachricht unsigniert aber HBCI-verschlüsselt an das Kundenprodukt übertragen.

		Das Kundensystem wertet nach der Entschlüsselung die Kreditinstitutsantwort aus und zeigt dem Kunden die Begrüßungsseite an.

b) Auftragseinreichung beim Secoder-Sicherheitsverfahren = 811

Bei diesem Szenario werden die SecCmds von der aktiven Kundenkomponente auf Basis des zugrunde liegenden Auftrags offline erzeugt und an den Secoder übertragen. Der Kunde fügt – nach Kontrolle und Bestätigung des Datenextraktes im Secoder-Display - eine fortgeschrittene Signatur und eine entsprechende Secodervisualisierungsbestätigungssignatur an. Beide werden zusammen mit dem Auftrag an das Institut gesendet.

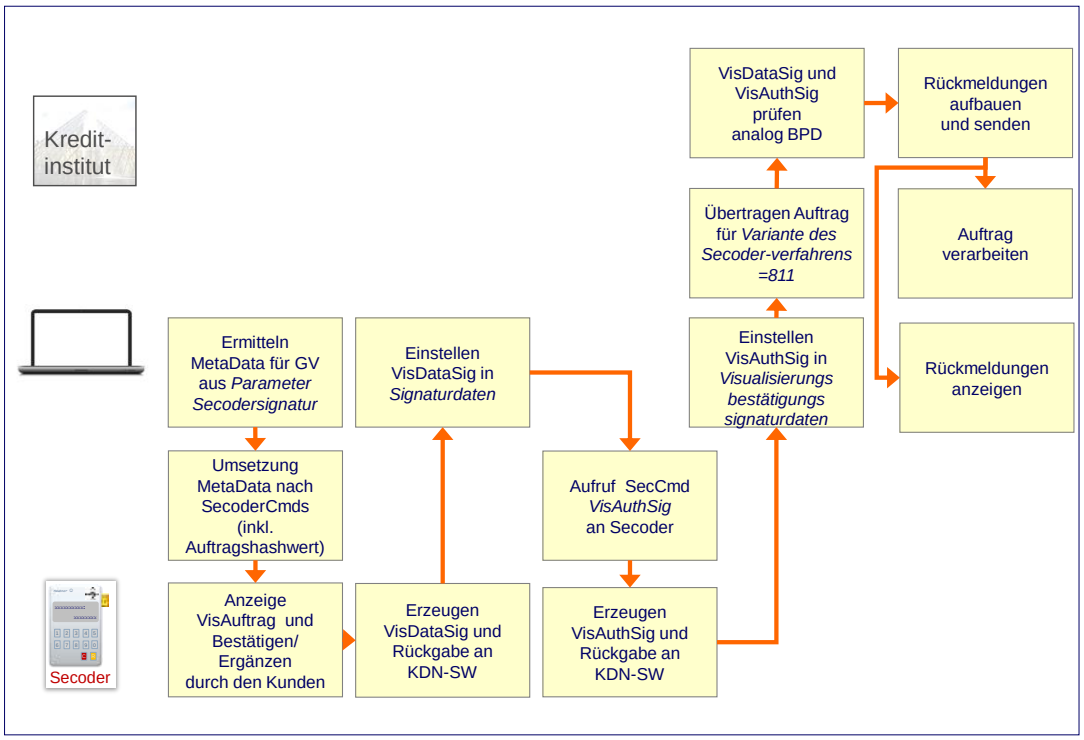


Abbildung 11: Fortgeschrittene Signatur mit Secoder-Sicherheitsverfahren=811

Auftragseinreichung bei Secoder-Sicherheitsverfahren=811 Ausgangszustand: • Die Initialisierung ist erfolgt; der Kunde hat dort durch entsprechende Belegung des Elementes *Secoder-Signatur* das Secoder-Sicherheitsverfahren 811 für die gesamte Kommunikation gewählt.		
Schritt 1a SEPASingRemitt_1_Req	→	Auftrag einreichen Im Kundenprodukt wird Auftragsnachricht für eine Einzelüberweisung *SingRemitt_1_Req* aufgebaut. Hierbei wird in Element *Signatur-ID* (in den *Secoder-spezifischen Parametern*) der zwischengespeicherte und um 2 (VisDataSig und VisAuthSig) inkrementierte Signaturzähler eingestellt.

		Das Kundenprodukt erzeugt auf Basis des Auftrags und der BPD die benötigten SecCmds und sendet diese zusammen mit dem Auftragshashwert über Signaturdaten und fachliches Segment an den Secoder im Applikationsmodus „aut“. Falls die Länge des Hashwerts nicht der Blocklänge entspricht, wird dieser mit Nullen aufgefüllt. Wie bei der Initialisierung beschrieben wird auch hier der VisData-Puffer sukzessive mit Informationen aus dem Secoder-Visualisierungsprozess gefüllt. Aus den Ergebnissen wird – mit dem zwischengespeicherten Hashwert über die Auftragsdaten startend – ein Hashwert über die Visualisierungsdaten gebildet, der den Inhalt des VisData-Puffers ersetzt. Über diesen Hashwert wird mit Hilfe des Signierschlüssels eine VisData-Signatur erzeugt. Die Kontrolle wird an das aufrufende Programm zurückgegeben und die VisData-Signatur in das Element *Signaturdaten* eingestellt. Das Kundenprodukt ruft den Secoder nochmals im Applikationsmodus „aut“ zur Visualisierungsbestätigung auf (VisAuthSig) und übergibt hierzu den in der BPD festgelegten Visualisierungsbestätigungssignaturfüllwert (Einzig möglicher Wert ist "SECODERSECODERSECODE"). Der noch im VisData-Puffer vorhandene Hashwert über die Secoder-Visualisierungsdaten wird an den ersten Stellen mit dem Visualisierungsbestätigungssignaturfüllwert überschrieben und über dieses Resultat eine VisAuth-Signatur gebildet. Die erzeugte VisAuth-Signatur wird an das Kundenprodukt zurückgegeben und dort in das Element *Visualisierungsbestätigungssignaturdaten* eingefügt. Der fachliche Geschäftsvorfall (z. B. *SingRemitt_1_Req*) wird HBCI-verschlüsselt zum Institut übertragen.
Schritt 1b z. B. Response	←	Rückmeldungen senden Nach erfolgreicher Entschlüsselung und Signaturprüfung kann der Auftrag verarbeitet werden. Es wird eine Auftragsantwort aufgebaut, die ggf. erzeugte Antwortsegmente sowie die Rückmeldungen zur Signatur-Prüfung und zum Auftrag selbst enthält Da bei Secoder-Sicherheitsverfahren=811 keine Institutssignatur verwendet wird, wird die Nachricht unsigniert aber HBCI-verschlüsselt an das Kundenprodukt übertragen. Das Kundensystem wertet nach der Entschlüsselung die Kreditinstitutsantwort aus.

III.2.3 Erweiterung der Rückmeldungscodes

Bei Verwendung des Secoder-Sicherheitsverfahrens können spezielle Rückmeldecodes vom Kreditinstitut zurückgemeldet werden, die rein Secoderverfahrensspezifisch sind und u. U. nicht direkt mit dem zugehörigen Geschäftsvorfall in Verbindung stehen. Eine Beschreibung aller Rückmeldungscodes befindet sich in [RMCode].

Es handelt sich hierbei um die folgenden Codes:

Warnungen und Hinweise

Code	Beispiel für Rückmeldungstext
3921	Zugelassene Secoder-Sicherheitsverfahren für den Benutzer (+ Rückmeldungsparameter)
3950 -999	Individuell

Fehlermeldungen

Code	Beispiel für Rückmeldungstext
9210	Auftrag abgelehnt – Auftragsdaten inkonsistent. Eingereichter Auftrag gelöscht
9210	Auftrag abgelehnt – Kompetenz nicht ausreichend
9330	Schlüsseleigner gesperrt
9330	Schlüssel gesperrt
9340	Signatur fehlerhaft
9340	Sicherheitsprofil unbekannt
9380	Gewähltes Secoder-Sicherungsverfahren nicht zulässig
9931	Sperrung des Kontos nach x Fehlversuchen
9931	Teilnehmersperre durchgeführt
9941	Signatur ungültig
9350	Zertifikat abgelaufen
9351	Zertifikat gesperrt
9352	Zertifikatseigner unbekannt
9953	Nur ein Signatur-pflichtiger Auftrag pro Nachricht erlaubt
9954	Mehrfach-Signaturen nicht erlaubt
9359	OCSP-Anfrage nicht beendet
9360	Signatur fehlerhaft – BPD nicht mehr aktuell

III.2.3.1 Beschreibung spezieller Rückmeldungen im Secoder-Sicherheitsverfahren

Rückmeldungscode 3921: Zugelassene Secoder-Sicherheitsverfahren für den Benutzer (+ Rückmeldungsparameter)

Der Rückmeldungscode 3921 dient dazu, dem Kundenprodukt im Rahmen der Initialisierungsantwort die für den Benutzer zugelassenen Secoder-Sicherheitsverfahren mitzuteilen. Hierzu werden in den Rückmeldungsparametern entsprechend den zugelassenen Verfahren („800“ bis „899“) maximal zehn mögliche Verfahren transportiert.

Das Kundenprodukt muss – unabhängig vom gewählten Secoder-Sicherheitsverfahren – bei jeder Initialisierung die vom Institut mit dem Rückmeldungscode 3921 übermittelten Werte für die Rückmeldungsparameter prüfen, gegen gespeicherte Informationen vergleichen und diese ggf. aktualisieren.
Sollte das Kundenprodukt in der Initialisierungsnachricht ein Verfahren wählen, das für den Benutzer nicht bzw. nicht mehr zugelassen ist, so beendet das Kreditinstitut die Kommunikation mit Rückmeldungscode 9800 in Kombination mit Code 3921 und meldet die aktuell zugelassenen Verfahren in den Rückmeldungsparametern zurück.

Rückmeldungscode 9360: Signatur fehlerhaft – BPD nicht mehr aktuell

Diese Rückmeldung bezieht sich speziell auf die Parametrisierung der Visualisierungsinformationen im Rahmen der Initialisierung. Es wird Instituts-seitig festgestellt, dass die Signaturprüfung fehlschlug, da das Kundensystem über keine aktuelle BPD verfügt. Die Kreditinstitutsnachricht wird analog den Informationen im Verschlüsselungskopf verschlüsselt und nicht signiert.

III.2.4 Parameterdaten Secodersignatur

Die Parameterdaten Secodersignatur (Tagname: *SecoderSignatureParm*) werden verwendet, um folgende Informationen zu beschreiben:

- Geschäftsvorfälle, die mit diesem Secoder-Sicherheitsverfahren ausgeführt werden dürfen, wie dies auch bei den anderen Sicherheitsverfahren der Fall ist (Tagname: *BusinessTransAllowed*).
- Visualisierungsinformationen pro Geschäftsvorfallgruppe (Tagname: *OrderSpecificVisualization*).

III.2.4.1 Generelles Secoder-Visualisierungskonzept

Das Visualisierungskonzept ist durch das Design des Secoders strikt vorgegeben. Im Speziellen gelten die Festlegungen der Secoder-Spezifikation [Secoder] und der „User Interface & Implementation Guide“ [Secoder Impl] inkl. der dort skizzierten Beispiele. Aus diesen Beispielen lassen sich für die Parametrisierung in der BPD drei grundsätzliche Anzeigedefinitionen ableiten (die Ausrichtung der Texte in der folgenden Abbildung sind exemplarisch).

Aufgrund der Komplexität der dezentralen Visualisierungsaufbereitung wird dringend empfohlen, dass die Secoder-Anwendungsfunktion, welche für die Umsetzung in die Secoder-Kommandos zuständig ist, über eine Trace-Funktion verfügt, um die Fehlersuche bei abweichenden Signaturergebnissen zu erleichtern.

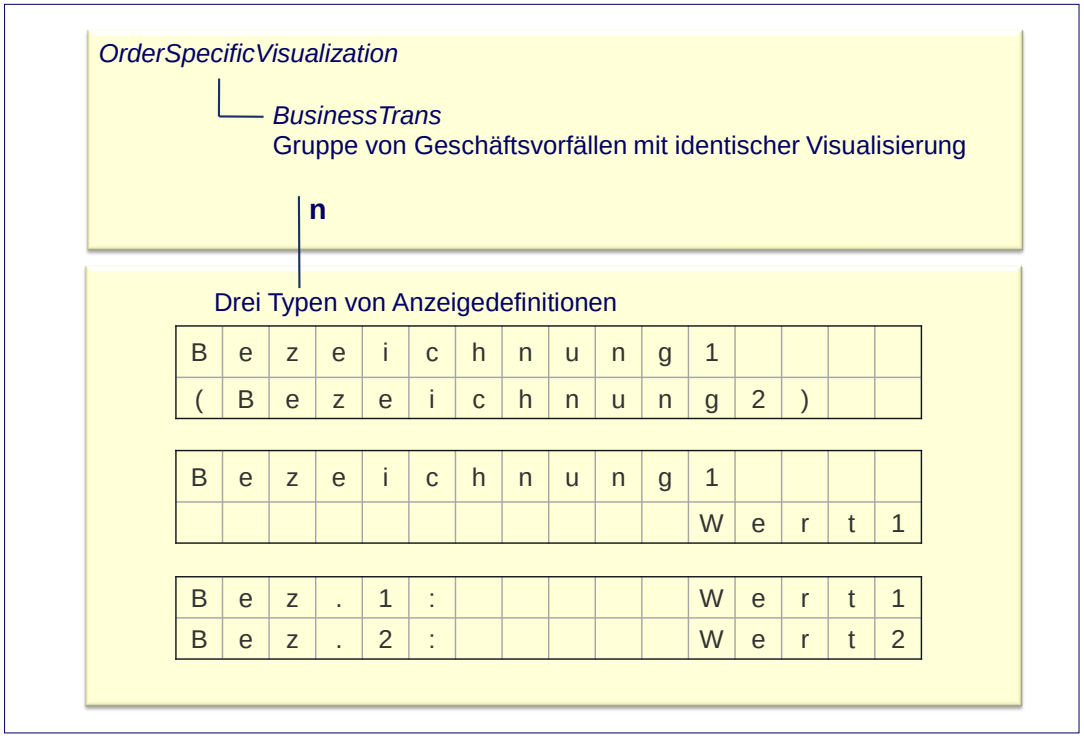


Abbildung 12: Struktur der geschäftsvorfallspezifischen Visualisierungsinformationen

In den folgenden Beispielen zur Visualisierung wird generell von einer heute marktüblichen Displaygröße von 2 x 16 Zeichen ausgegangen, wobei die Physik in den *SecoderSignatureParam* keine Rolle spielt, sondern diese durch die Secoder-Anwendungsfunktion an die physischen Eigenschaften des verwendeten Secoders angepasst wird. Ein sog. „Dataset (DS)“ (Tagname: *SecoderDataset*, Begriff aus der Secoder-Spezifikation) umfasst eine Zeile, d. h. 1 x 16 Zeichen.

Pro Geschäftsvorfall (bzw. in FinTS: Geschäftsvorfallgruppe mit identischer Visualisierung) können – abhängig vom zur Verfügung stehenden VisData-Pufferbereich im Secoder theoretisch bis zu 255 solcher Datasets verwendet werden.

Die drei in Abbildung 12 gezeigten typischen Anzeigedefinitionen haben folgende Bedeutung:

1. Die erste Anzeigedefinition dient der Darstellung von einer oder zwei textuellen Bezeichnungen, die linksbündig dargestellt werden, z. B. die Texte „Einzelüberweis.“ und „Inland“. Als Sonderfall dieser Anzeigedefinition kann die zweite Displayzeile auch leer bleiben.

2. Die zweite Anzeigedefinition kombiniert eine Bezeichnung (z. B. den Text „Ziel-IBAN“) in Zeile 1 und einen zugehörigen Wert (z. B. die IBAN des Begünstigten aus einem SEPA-Format) in Zeile 2.
3. Mit der 3. Anzeigedefinition ist es möglich, je Zeile eine Bezeichnung und einen Wert darzustellen.

Die Variante 3 sollte vor dem Hintergrund unterschiedlicher Displaygrößen (z. B. 2 x 16, 4 x 16, 2 x 32) bei Marktprodukten nur ganz gezielt eingesetzt werden, wenn sichergestellt werden kann, dass die Darstellung auf allen zu unterstützenden Displays gut lesbar ist und keine falschen Interpretationen erlaubt .

Auf Basis dieser drei Anzeigedefinitionen ist es möglich, sieben grundsätzliche Szenarien abzubilden, die i. W. den Beispielen aus dem User Interface & Implementation Guide [Secoder Impl] entsprechen. Die Anzeigedefinitionen verfügen über folgenden Befehlsvorrat[2]:

Secoder-visualisierung Referenz	Verweis auf den jeweiligen Secodervisualisierungstext. z. B. *BankParamData/SecurityMethodParam/ SupportedMethod/Secoder/SecoderSignatureParam/ OrderSpecificVisualization/OrderVisualization/TextReference* Default: Keiner
Display-Position	Ausrichtung des Dataset im Secoder-Display (Links, Rechts[3]). Diese Angabe ist nötig, da die Secoder-Texte in der BPD nicht gepadded werden. Default: Links
Länge (Secoder-Text)	Länge des Secoder-Textes Default: 0
Secoder-Text	Secoder-Text, entweder statisch durch die Anzeigedefinition selbst oder dynamisch aus den Auftragsdaten oder leer. Default: leer
Länge Secoder-Eingabedaten	Soll ein Wert nicht nur bestätigt, sondern komplett eingegeben oder ergänzt werden, wird hierdurch die Länge der geforderten Eingabedaten am Secoder vorgegeben. Default: 0
Ausrichtung und Format Secoder-Eingabedaten	Dieser Parameter beschreibt die Ausrichtung bei der Eingabe der Daten (Textmodus oder Taschenrechnermodus) und die Verwendung des Kommas. Default: Textmodus, falls Länge (Secoder-Text) > 0

[2] Aus Gründen der besseren Lesbarkeit – vor allem in den Abbildungen – werden Begriffe wie *Secodervisualisierung Referenz* auf *Referenz* gekürzt. Die exakten Namen befinden sich im Data Dictionary unter *Secodervisualisierungstexte*.

[3] Die im Secoderkonzept theoretisch mögliche Ausprägung „Mitte“ wird in FinTS nicht benutzt.

	Taschenrechnermodus, sonst
Secoder-Padding	Dieser Parameter beschreibt das Padding der Daten im Vis-Data-Puffer des Secoders. Default: E0 bei numerischen Daten EF bei alfanumerischen Daten

Nähere Festlegungen zum konkreten Mapping zwischen den FinTS Visualisierungsparametern und den Secoder-Kommandos befinden sich in Abschnitt III.2.4.2.

Die Secodervisualisierungstexte (Tagname <SecoderVisualizationParams>) beschreiben die Anzeigedefinitionen auf die bei der Beschreibung der Datasets referenziert werden kann.

Die Secodervisualisierung MetaData (*Geschäftsvorfallspezifische Visualisierungsinformationen*) enthalten die Definitionen pro Geschäftsvorfallgruppe.

Pro Geschäftsvorfallgruppe wird über Pfade auf die entsprechenden Secodervisualisierungstexte referenziert, wobei berücksichtigt werden muss, ob eine Abfrage oder Bestätigung ein- oder zweizeilig dargestellt wird.

Da in den Secodervisualisierungstexten im Element *SecoderDataset* auch Variable – dargestellt durch den Platzhalter „#“ – enthalten sind, befinden sich in den Geschäftsvorfallgruppenspezifischen Visualisierungsinformationen (Tagname *Ordervisualization*) die jeweiligen Werte hierfür (z. B. konkrete IBAN aus FinTS, DTA, SEPA …).

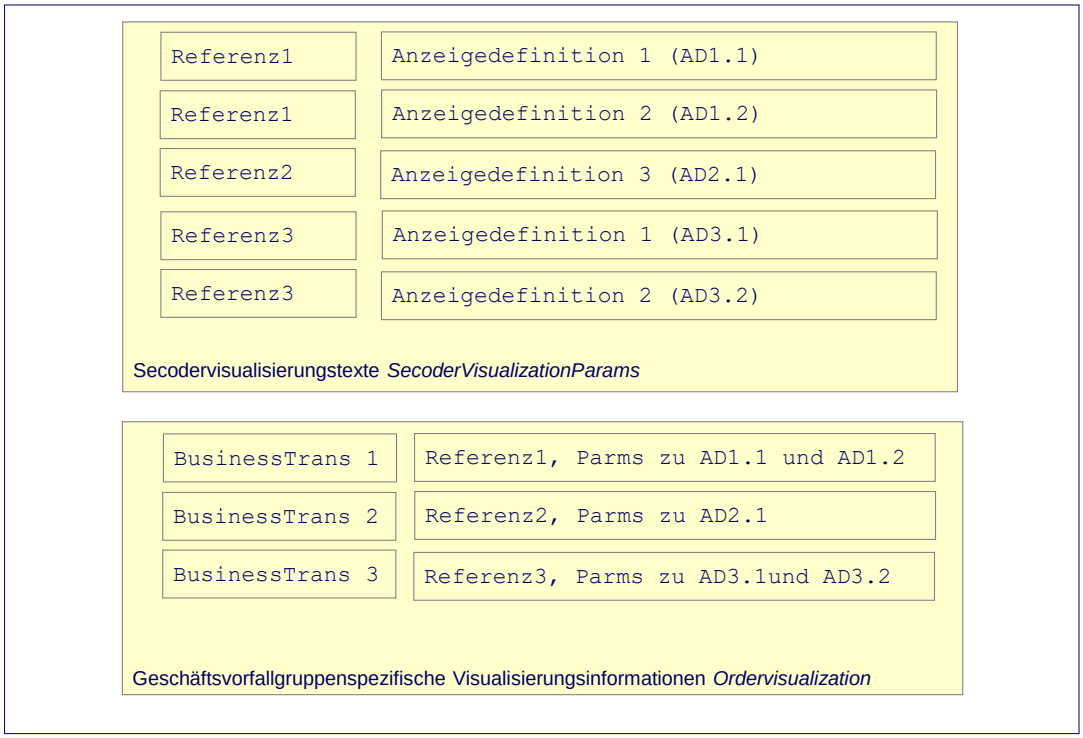


Abbildung 13: Zusammenhang zwischen Secoder MetaData und Secodervisualisierungstexten

III.2.4.1.1 Element „Ausrichtung und Format von Secoder-Eingabedaten“

Die Ausrichtung der Secoder-Eingabedaten kann auf zwei Arten erfolgen:

a) Textmodus

Im Textmodus beginnt die Eingabe direkt hinter dem letzten Zeichen des Secoder-Textes, d. h. der Cursor blinkt an der ersten Stelle rechts vom Secoder-Text. Ist der Secoder-Text leer, blinkt der Cursor im verwendeten Beispiel an Stelle 16 ganz rechts.

Jedes eingegebene Zeichen schiebt den Cursor um eine Position nach rechts, bis die vorgegebene Anzahl von Eingabezeichen erreicht ist. Der Secoder-Text und die bereits eingegebenen Zeichen werden nicht bewegt.

Der Textmodus stellt die Default-Belegung dar, wenn ein Secoder-Text vorhanden ist.

b) Taschenrechnermodus

Im Taschenrechnermodus blinkt der Cursor grundsätzlich im verwendeten Beispiel an Position 16 der Zeile ganz rechts, auch wenn ein Secoder-Text vorhanden ist. Jedes eingegebene Zeichen schiebt die bereits vorhandenen Zeichen (ggf. Secoder-Text und Eingabezeichen) um eine Position nach links.

Der Taschenrechnermodus stellt die Default-Belegung dar, wenn kein Secoder-Text vorhanden ist.

III.2.4.1.2 Repräsentative Beispiele zum Secoder-Visualisierungskonzept

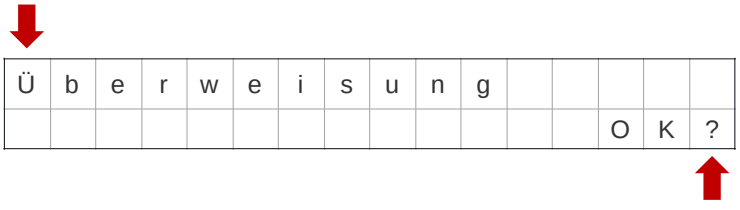
Die folgenden Kapitel beschreiben die Parametrisierung anhand von sieben repräsentativen Konstellationen, wie sie auch im User Interface & Implementation Guide [Secoder Impl] verwendet werden. Das Mapping der FinTS-Definitionen in die SecCmds ist mit der dort beschriebenen Syntax der Secoder Data Confirmation vorzunehmen.

Bei der Sicherheitsfunktion 811 kann nur eine Bestätigung von Daten, kein Auffüllen erfolgen.

Bei den Beispielen wird wie in [Secoder_Impl] von einer Displaygröße von 2 x 16 Zeichen ausgegangen.

a) Szenario 1: Bestätigung statischer Werte, 2-zeilig

Szenario 1 dient zur Darstellung statischer Werte im Display des Secoders. Die Anzeigedefinition lässt die Belegung beider Displayzeilen mit je einem maximal 16-stelligen Textstring zu. Die Bezeichnungen werden ohne Leerzeichen eingestellt. Über die Parametrisierung im Element *Display-Position* wird eine entsprechende Ausrichtung erreicht. In den Beispielen werden wo immer möglich die Defaultwerte verwendet, um eine möglichst kompakte BPD-Modellierung zu erzielen.



Aufbau der beiden zugehörigen Visualisierungstexte:

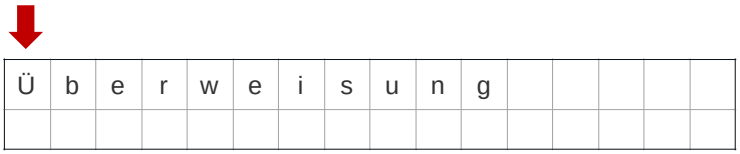
Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
1	**Def**	**Def**	**„Überweisung“**	**Def**	**Def**	**Def**

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
1	**R**	**Def**	**„OK?“**	**Def**	**Def**	**Def**

Die Bezeichnungen werden ohne Leerzeichen eingestellt. Über die Parametrisierung im Element *Display-Position* wird eine entsprechende Ausrichtung erreicht. In den Beispielen werden wo immer möglich die Defaultwerte verwendet, um eine möglichst kompakte BPD-Modellierung zu erzielen.

b) Szenario 2: Bestätigung statischer Werte, 1-zeilig

Szenario zwei beschreibt die Darstellung eines 1-zeiligen Textes im Secoder-Display. Der Kunde geht in diesem Fall ohne weitere Benutzerführung davon aus, dass er diesen Text bestätigen muss, bevor die Verarbeitung im Secoder fortgesetzt werden kann.



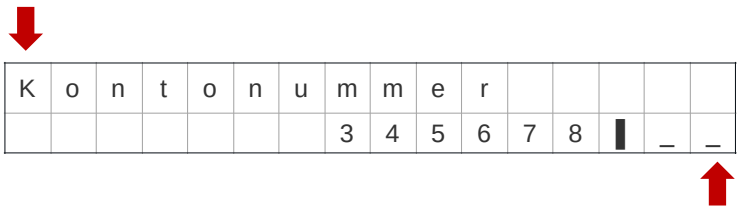
Aufbau des zugehörigen Visualisierungstextes:

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
2	**Def**	**Def**	**„Überweisung“**	**Def**	**Def**	**Def**

Zu beachten ist, dass in FinTS nur die Secoderoption durch Wegfall des Dataset für die zweite Zeile unterstützt wird, nicht die Möglichkeit ein Dataset mit Länge 0 zu verwenden (dies würde zu einem unterschiedlichen Kryptogramm führen).

c) Szenario 3: Auffüllen mit Ziffern, 2-zeilig

In Szenario 3 wird in Zeile 1 ein statischer Text angezeigt. Zeile 2 enthält einen Teil der Kontonummer (des Begünstigten), die letzten 3 Stellen muss der Kunde ergänzen, bevor er den gesamten Ausdruck bestätigen kann.



Aufbau der beiden zugehörigen Visualisierungstexte:

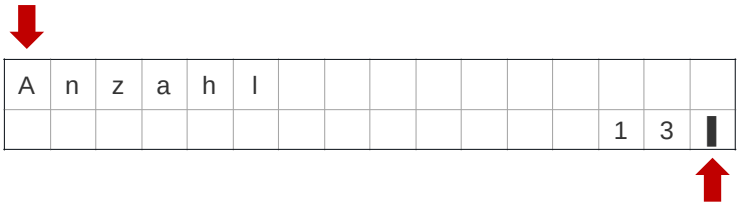
Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
3	**Def**	**Def**	**„Kontonummer“**	**Def**	**Def**	**Def**

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
3	**R**	**6**	**#**	**3**	**Def**	**Def**

In Zeile 1 wird der statische Text aus dem Element *Secoder-Text* angezeigt. Zeile 2 besteht aus einem dynamischen Text, was durch den Platzhalter „#“ im Element *Secoder-Text* dargestellt ist. Der konkrete Wert für den Platzhalter wird durch die *Geschäftsvorfallspezifische Visualisierungsinformation* festgelegt.

d) Szenario 4: Eingabe von Ziffern, 2-zeilig

Szenario 4 ist dadurch gekennzeichnet, dass der Wert in Zeile 2 komplett vom Kunden eingetippt werden muss.



Aufbau der beiden zugehörigen Visualisierungstexte:

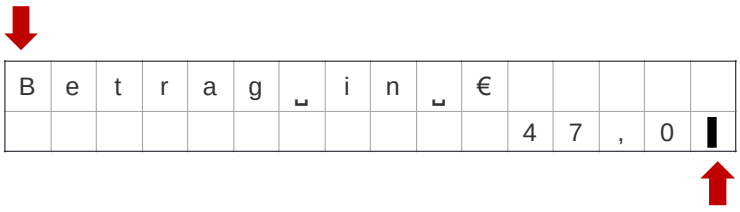
Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
4	**Def**	**Def**	**„Anzahl“**	**Def**	**Def**	**Def**

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
4	**R**	**Def**	**Def**	**3**	**Def**	**Def**

In Zeile 2 wird die geforderte Länge der Eingabedaten angegeben. Alle anderen Werte entsprechen der Default-Belegung.

e) Szenario 5: Eingabe von Komma-Zahlen, 2-zeilig

Szenario 5 verwendet als Eingabe eine Kommadarstellung.



Aufbau der beiden zugehörigen Visualisierungstexte:

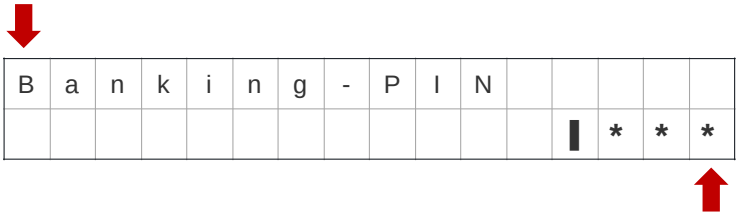
Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
5	**Def**	**Def**	**„Betrag in €“**	**Def**	**Def**	**Def**

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
5	**R**	**Def**	**Def**	**5**	**05**	**Def**

In Zeile 2 wird zusätzlich zur geforderten Datenlänge 5 (inklusive Komma) das Formatkennzeichen „05“ für die Darstellung von 2 Kommastellen am Secoder verwendet.

f) Szenario 6: Verdeckte Eingabe, 2-zeilig

In Szenario 6 wird die Verwendung der verdeckten Eingabe angewendet. Die Eingabe der Banking-PIN ist mit Sicherheitsfunktion 811 auf diese Weise nicht unterstützt, da die Applikation „aut“ nicht über eine zentrale Anwendungs-PIN, sondern über die CSA-PIN geschützt ist, die lokal geprüft wird.



Aufbau der beiden zugehörigen Visualisierungstexte:

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
6	**Def**	**Def**	**„Banking-PIN“**	**Def**	**Def**	**Def**

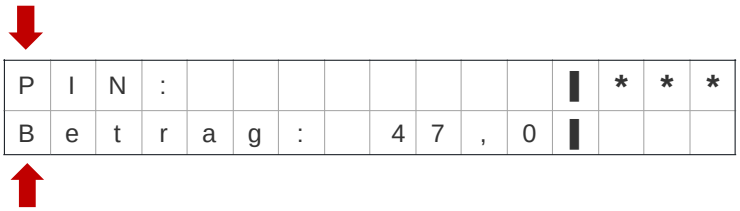
Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
6	**R**	**Def**	**Def**	**4**	**04**	**Def**

Die verdeckte Eingabe wird durch das Kennzeichen „04“ für die Darstellung am Secoder erreicht.

g) Szenario 7: Auffüllen mit Ziffern in einer Zeile

Szenario 7 stellt eine kompakte Darstellung der Eingabe von zwei Werten inklusive Beschreibung in je einem Dataset dar.

Hinweis: Diese Darstellung kann aus Sicht der Benutzerfreundlichkeit ungünstig wirken, wenn Secoder-Produkte mit unterschiedlichen Displaygrößen, z. B. 2 x 32, in identischer Weise angesteuert werden sollen.



Aufbau der beiden zugehörigen Visualisierungstexte:

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
7	**Def**	**12**	**„PIN:“**	**4**	**04**	**Def**

Idx	Dis-play-Pos	Länge (Secoder-Text)	Secoder-Text	Länge Sec-Eingabe-Daten	Ausrichtung und Fmt. Sec-Eingabedaten	Secoder-Padding
7	**Def**	**8**	**„Betrag:“**	**5**	**05**	**Def**

Die Verwendung von Secoder-Text und Eingabe in einer Zeile wird durch die Kombination eines statischen Secoder-Textes „Betrag:“, der durch die Längenangabe „8“ mit Leerzeichen ergänzt wird, mit einer 5-stelligen Eingabe erreicht.

Restriktion:
Als Einschränkung bei diesem Szenario gilt, dass keine Kombination eines statischen Secoder-Textes (im Beispiel „Betrag:“) mit einem dynamischen Textfragment (z. B. aus der DTA-Definition), das dann noch manuell zu ergänzen wäre, möglich ist. Hierfür muss eine zweizeilige Darstellung pro Parameter gewählt werden.

Szenario 7 sollte vor dem Hintergrund unterschiedlicher Displaygrößen (z. B. 2 x 16, 4 x 16, 2 x 32) bei Marktprodukten nur ganz gezielt eingesetzt werden, wenn sichergestellt werden kann, dass die Darstellung auf allen zu unterstützenden Displays gut lesbar ist und keine falschen Interpretationen erlaubt .

III.2.4.2 Struktur der Visualisierungsdaten

Die Parameterdaten Secodersignatur (Tagname: *SecoderSignatureParam*) haben folgenden Aufbau:

- Secoder Visualisierungstexte
  (Tagname: *SecodervisualizationParams*)
- Geschäftsvorfallspezifische Visualisierungsinformationen für Secoder
  (Tagname *OrderSpecificVisualization*)

Die generelle Struktur wird aus folgendem Diagramm ersichtlich:

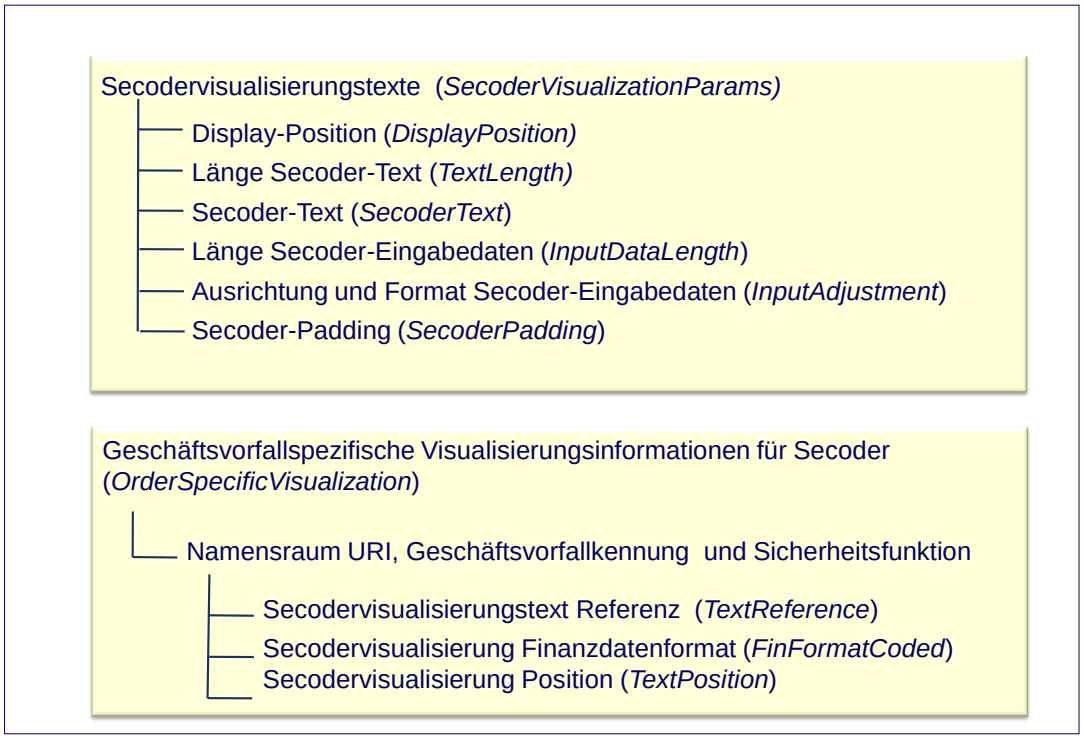


Abbildung 14: Struktur der Visualisierungsdaten

III.2.4.3 Tabelle der Secodervisualisierungstexte

Im Display des Secoders können zunächst beliebige Texte angezeigt werden. Um jedoch Redundanzen zu vermeiden und die Visualisierungsinformationen in den BPD möglichst kompakt zu halten, werden die verwendeten Texte und die Secoder MetaData in einer Tabelle jeweils bestehend aus Referenz, zugehörigem Text und MetaData zusammengefasst:

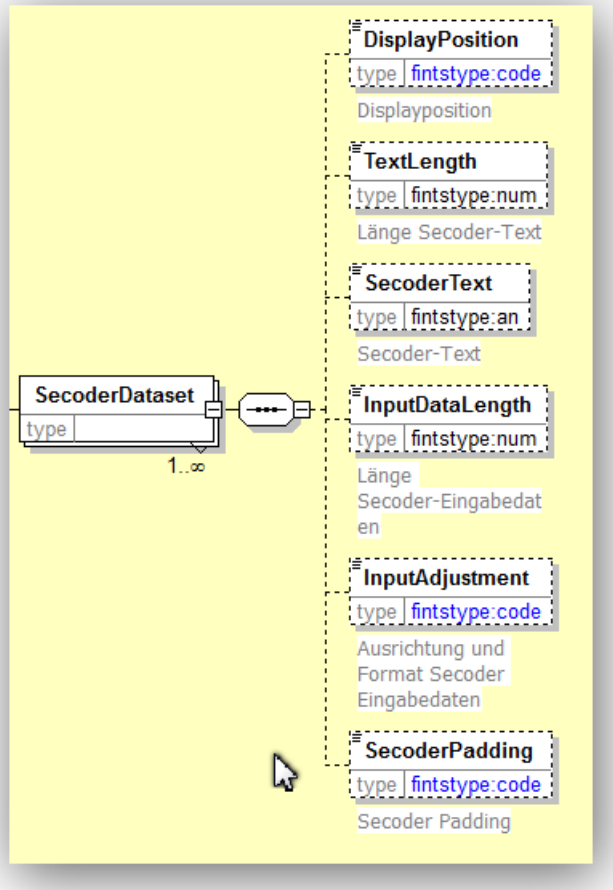


Abbildung 15: Tabelle der Secodervisualisierungstexte und Secoder MetaData

Die einzelnen Secodervisualisierungstexte können so über die Angabe der Referenz in den geschäftsvorfallspezifischen Visualisierungsinformationen referenziert werden. Es können nur Texte dargestellt werden, die auch in der Tabelle der Secodervisualisierungstexte enthalten sind.

Die Secoder MetaData sind vom Funktionsumfang so gestaltet, dass sie 1:1 in SecCmds umgesetzt werden können, wie sie im Secoder User Interface & Implementation Guide [Secoder Impl] beschrieben sind.

Im Einzelnen sind folgende Werte möglich:

Secoder-visualisierung Index

Referenz des jeweiligen Secodervisualisierungstextes, die in den „geschäftsvorfallspezifischen Visualisierungsinformationen“ *OrderSpecificVisualization* referenziert wird. Da im angenommenen Beispiel ein Dataset 1 x 16 Zeichen umfasst, werden 2 Anzeigedefinitionen benötigt, um z. B. 2 x 16 Zeichen für die Darstellung von Bezeichnung und Wert eines Auftragsbestandteils am Secoder darzustellen. Hierbei weisen beide Anzeigedefinitionen dieselbe Referenz auf. Die Reihenfolge der Anzeigedefinitionen in der Tabelle der Secodervisualisierungstexte entspricht der Reihenfolge der Anzeige dieser Texte am Secoder.

Default: Keiner

Display-Position

Ausrichtung des Dataset im Display. Diese Angabe ist nötig, da die Secoder-Texte in der BPD nicht gepadded werden.

Mögliche Werte sind:

L: Links
R: Rechts

Die Secoder-Anwendungsfunktion kann über die Einstellung von DS_X.L-VIS und der realen Displaygröße die gewünschte Ausrichtung einstellen.

Default: Links

Länge (Secoder-Text)

Folgende Möglichkeiten existieren:

1. Statischer Text, z. B. „Ziel-IBAN:“
   Hier ergibt sich die Länge implizit aus der Länge des Secoder-Textes und muss nicht angegeben werden (Default).
2. Dynamischer Wert, z. B. „12345678“
   Die Daten werden dynamisch aus dem Auftrag (z. B. SEPA-Format) übernommen und werden in der angegebenen Länge verwendet. In diesem Fall muss ein konkreter Wert für die Länge angegeben werden.

Default: 0

Secoder-Text

Für diesen Parameter existieren drei Ausprägungen:

1. Statischer Text
   die Bezeichnung („Label“) des zu bestätigenden bzw. zu ergänzenden Wertes, z. B. „Ziel-IBAN:“.
2. Fester Anteil des dynamischen Werts
   Der zu bestätigende Wert selbst in Form des Platzhalters # bzw. dessen fester Anteil, z. B. „12345678“ bzw. „12345___“.
3. Leer
   Es ist weder ein statischer Text noch ein fester Anteil eines dynamischen Wertes angegeben, d. h. der Kunde muss den Wert komplett eingeben.

Statische Secoder-Texte stehen fest und ohne Padding in

der BPD. Dynamische Werte (komplett oder anteilig) werden durch den Platzhalter „#“ repräsentiert. Pro Anzeigedefinition, die aus bis zu zwei Datasets mit identischem Index bestehen kann, kann maximal ein Platzhalter „#“ definiert werden.

Default: leer

Länge Secoder-Eingabedaten

Soll ein Wert nicht nur bestätigt, sondern komplett eingegeben oder ergänzt werden, wird hierdurch die Länge der geforderten Eingabedaten am Secoder vorgegeben.

Hinweis: Das Ergänzen von Daten kann bei der Sicherheitsfunktion 811 nicht genutzt werden; es kann nur eine Bestätigung von Daten erfolgen.

Default: 0

Ausrichtung und Format Secoder-Eingabedaten

Dieser Parameter beschreibt die Ausrichtung bei der Eingabe der Daten und die Verwendung des Kommas. Er entspricht dem Secoder-Parameter DS_x.DCF (Details hierzu siehe unter [Secoder_Impl]) und hat folgende Ausprägungen:

00	Eingabeposition ist ab der ersten Stelle hinter dem Secoder-Text bzw. rechts, falls kein Secoder-Text vorhanden.
01	Die Eingabe startet rechts (Taschenrechnermodus).
04	Wie 00, jedoch werden die Eingaben verdeckt dargestellt. Dieses Format ist bei der Sicherheitsfunktion 811 nicht zugelassen.
03 05 07	Wie 01, jedoch mit 1, 2 oder 3 Kommastellen.

Default:
00, falls Länge (Secoder-Text) > 0
01, sonst

Secoder-Padding

Dieser Parameter beschreibt das Padding der Daten im Vis-Data-Puffer des Secoders und entspricht dem Secoder-Parameter DS_x.VCI (Details hierzu siehe unter [Secoder_Impl]) und hat folgende Ausprägungen:

D0	Numerische Daten (BCD) mit 0-Padding
E0	Numerische Daten (BCD) mit F-Padding
DF	Alfanumerische Daten (ISO 646) mit 0-Padding
EF	Alfanumerische Daten (ISO 646) mit F-Padding

Default:
E0 bei numerischen Daten
EF bei alfanumerischen Daten

Es bestehen die folgenden Zusammenhänge zwischen den MetaData-Definitionen und dem Aufbau des Secoder Data Confirmation Kommandos.

BPD-Parameter	Secoder-Kommando
Referenz	keine Entsprechung
Display-Position	keine Entsprechung, wird anhand $DS_x.L_{VIS}$ und physischer Displaygröße umgesetzt
keine Entsprechung, abgeleitet aus Text	$DS_x.L_{DB}$
Secoder-Text	$Dataset_x.Datablock_x (DS_x.DB_x)$
Länge Secoder-Eingabedaten	$DS_x.L_{DB}$ + L (Secoder-Eingabedaten) = $DS_x.L_{VIS}$
Ausrichtung und Format Secoder-Eingabedaten	DS_x.DCF
Ziffern/Buchstaben	DCF = ´00´ / ´01´
Kommastellen K1 bis K3	DCF = ´03´/ ´05´/ ´07´
Asterisk	DCF = ´04´
Secoder-Padding	DS_x.VCI

Abbildung 16: Analogien zwischen MetaData und Secoder Data Confirmation

III.2.4.4 Geschäftsvorfallspezifische Visualisierungsinformationen für Secoder

Die Mechanismen zur Steuerung des Visualisierungsvorgangs werden durch die folgende Abbildung verdeutlicht. Dabei sind die Werte für Segmentversion und Secoder-Sicherheitsverfahren mit „0“ als Default definiert, um die Anzahl der Einträge zu minimieren. Werden hierbei explizite Werte benutzt, um unterschiedliche Visualisierungen zu erreichen (z. B. Namensraum URI‘s und Geschäftsvorfallkennungen und/oder Secoder-Sicherheitsverfahren=811), so muss pro Ausprägung eine eigene Definition vorhanden sein.

Hinweise:

1. Alle nicht explizit modellierten Segmentversionen bzw. Secoder-Sicherheitsverfahren werden analog dem Default-Eintrag behandelt. Beim aktuellen Stand der Spezifikation mit nur einem Secoder-Sicherheitsverfahren ist dies gleichbedeutend mit dem Default-Eintrag.
2. Für jedes in den Secoder Parameterdaten angegebene Secoder-Sicherheitsverfahren muss eine Definition (Default oder Explizit) vorhanden sein.

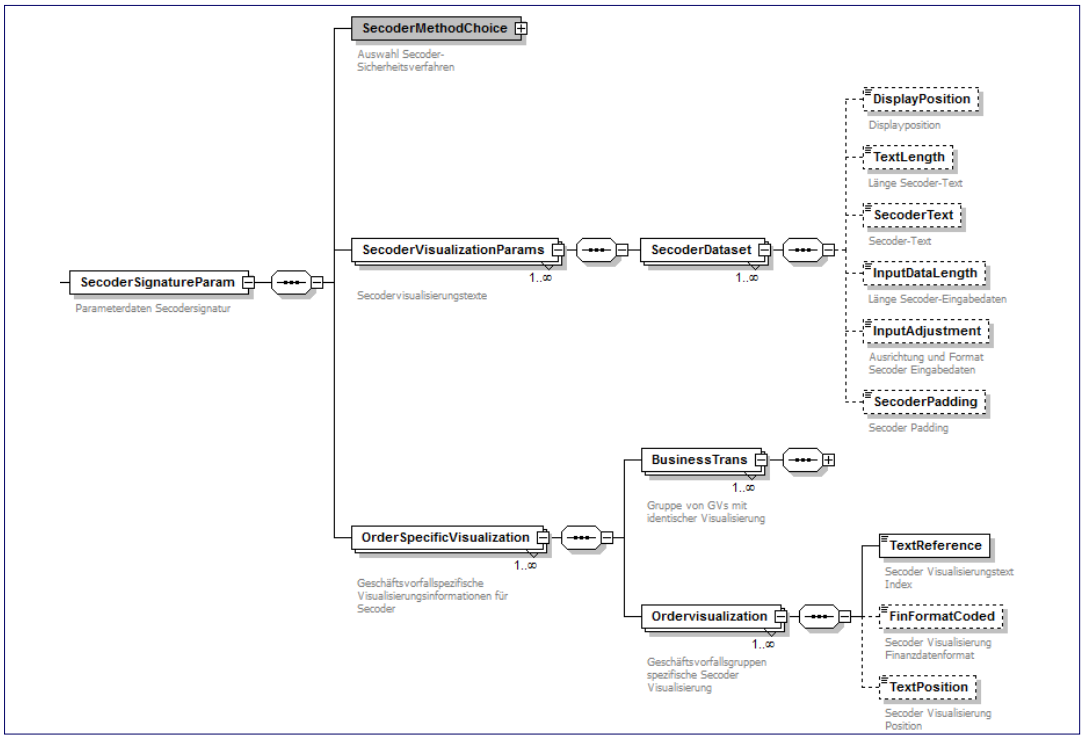


Abbildung 17: Definition der Secoder MetaData pro Geschäftsvorfall

Das Parametersegment enthält im Abschnitt der Geschäftsvorfallspezifischen Visualisierungsinformationen pro Anzeigendefinition einen Eintrag mit folgendem Aufbau:

Referenz	Referenz auf den / die darzustellenden Visualisierungstext(e) in der Tabelle der Secodervisualisierungstexte. Die dort enthaltenen Texte (1 bis n Datasets) werden in der definierten Reihenfolge abgearbeitet.
Finanzdatenformat	Kennzeichnung der Art des Finanzdatenformates wie z. B. FinTS, DTA oder SEPA; dies ist Voraussetzung für die Positionierung innerhalb eines Finanzdatenformates, um die zu visualisierenden Daten zu lokalisieren. Außer FinTS sind dort alle vorkommenden Finanzdatenformate definiert.
Position	Je nach Finanzdatenformat wird die Position des zu visualisierenden Elementes angegeben, z. B. FinTS[4] DTA: E.4 → 4. Element im E-Satz SEPA[5]

[4] Beispiel: `SingRemitt_1_Req/PayeeAcct/AcctNo`
[5] Beispiel: `SEPASingRemitt_1_Req/SEPAPainMsg/pain.001.001.02/GrpHdr/CtrlSum`

Je Anzeigedefinition kann maximal ein Positionsparameter festgelegt werden.

Dieser ersetzt dann den Platzhalter „#“ an der entsprechenden Stelle des jeweiligen Secodervisualisierungstextes.

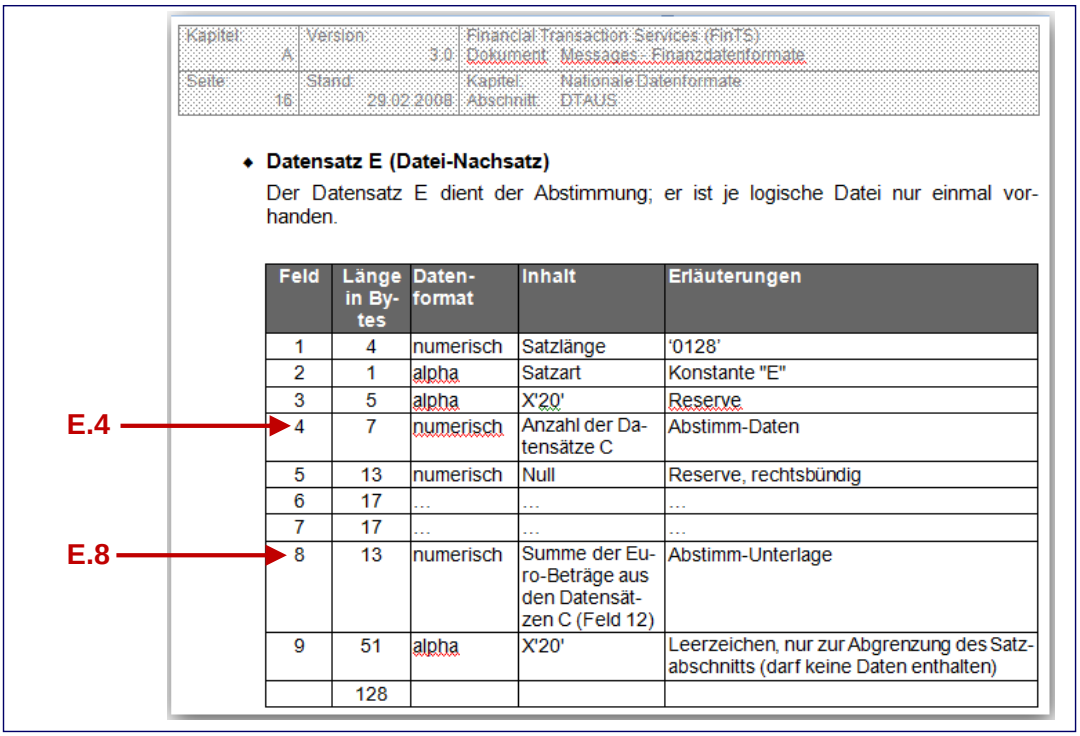
III.2.4.5 Positionierung bei der Secodervisualisierung

Abhängig vom jeweiligen Finanzdatenformat existieren unterschiedliche Adressierungsmöglichkeiten für die einzelnen Elemente im Format. Es handelt sich dabei im Regelfall um die Auswertung der Kundennachrichten, welche ein oder mehrere Finanzdatenformate enthalten können, d. h. ein Kundenprodukt kann – ‚offline‘ – vor der Einreichung des Auftrags auf Basis der BPD die Secodervisualisierungsdaten ermitteln. Gleiches gilt für Werte, die im Rahmen der Initialisierung visualisiert werden können, wie z. B. die Benutzerkennung.

Enthält die Kundennachricht mehrere Aufträge – in Form mehrerer FinTS-Elemente oder Sammelaufträge – so muss beim Aufbau der BPD darauf geachtet werden, dass ein Visualisierungselement ausgewählt wird, das sich auf alle Aufträge bezieht wie z. B. Summenwerte oder ein bestimmtes Auftreten eines Wertes im Auftrag. Iterationen von Visualisierungselementen über enthaltene Einzelaufträge – also die Anzeige mehrerer Instanzen eines Wertes im Secoder – sind nicht vorgesehen.

Da durch die Definition im Element „Secodervisualisierung Finanzdatenformat“ (Tagname: *FinFormatCoded*) eine Festlegung nur für ein Secoder-Visualisierungselement gemacht wird, können die Visualisierungsdaten für einen Auftrag auch aus unterschiedlichen Finanzdatenformaten bestehen, z. B. aus ggf. vorangestellten FinTS-Datenelementen und einem SEPA-Format.

III.2.4.5.1 Positionierung bei DTA



Kapitel: A | Version: 3.0 | Financial Transaction Services (FinTS)
Dokument: Messages - Finanzdatenformate
Seite: 16 | Stand: 29.02.2008 | Kapitel: Nationale Datenformate
Abschnitt: DTAUS

◆ **Datensatz E (Datei-Nachsatz)**

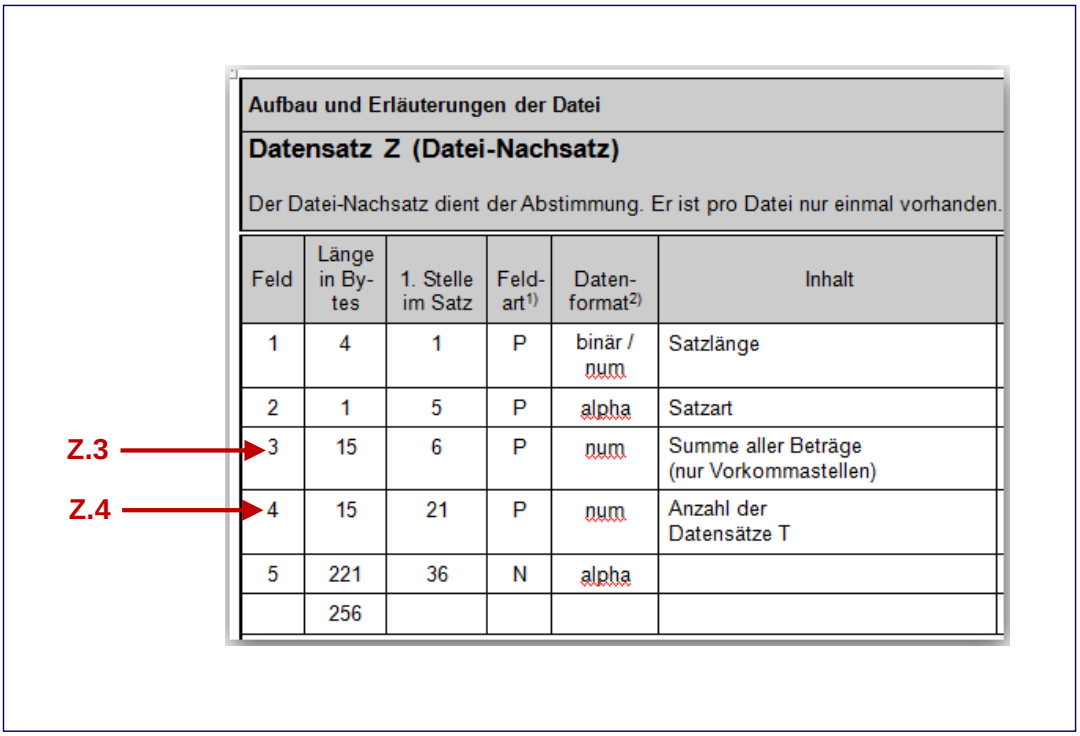
Der Datensatz E dient der Abstimmung; er ist je logische Datei nur einmal vorhanden.

Feld	Länge in Bytes	Datenformat	Inhalt	Erläuterungen
1	4	numerisch	Satzlänge	'0128'
2	1	alpha	Satzart	Konstante "E"
3	5	alpha	X'20'	Reserve
4	7	numerisch	Anzahl der Datensätze C	Abstimm-Daten
5	13	numerisch	Null	Reserve, rechtsbündig
6	17	...	...	...
7	17	...	...	...
8	13	numerisch	Summe der Euro-Beträge aus den Datensätzen C (Feld 12)	Abstimm-Unterlage
9	51	alpha	X'20'	Leerzeichen, nur zur Abgrenzung des Satzabschnitts (darf keine Daten enthalten)
	128			

Abbildung 18: Adressierung der Secodervisualisierungsdaten bei DTA-Formaten

Beim Format „DTA“ wird durch die erste Stelle der Datensatz innerhalb des DTA-Formates festgelegt; die zweite Stelle bezeichnet das Feld innerhalb des DTA-Datensatzes. Es erfolgt also grundsätzlich eine zweistufige Adressierung.

III.2.4.5.2 Positionierung bei DTAZV



Aufbau und Erläuterungen der Datei

Datensatz Z (Datei-Nachsatz)

Der Datei-Nachsatz dient der Abstimmung. Er ist pro Datei nur einmal vorhanden.

Feld	Länge in Bytes	1. Stelle im Satz	Feldart[1)]	Datenformat[2)]	Inhalt
1	4	1	P	binär / num	Satzlänge
2	1	5	P	alpha	Satzart
3	15	6	P	num	Summe aller Beträge (nur Vorkommastellen)
4	15	21	P	num	Anzahl der Datensätze T
5	221	36	N	alpha	
	256				

Abbildung 19: Adressierung der Secodervisualisierungsdaten bei DTAZV-Formaten

Beim Format „DTAZV“ wird durch die erste Stelle der Datensatz innerhalb des DTAZV-Formates festgelegt; die zweite Stelle bezeichnet das Feld innerhalb des DTAZV-Datensatzes. Es erfolgt also grundsätzlich eine zweistufige Adressierung.

III.2.4.5.3 Positionierung bei XML Formaten (FinTS, SEPA, camt)

Bei XML-Formaten wie FinTS oder SEPA wird zur Bezeichnung der im Secoder-Display zu visualisierenden Daten ein entsprechender XPATH-Ausdruck verwendet.

BEISPIEL FOLGT IM NÄCHSTEN DRAFT

Hinweis: für die Visualisierung der SEPA-Formate können auch die Informationen aus den Datenelementen des FinTS-Geschäftsvorfalls (außerhalb des eingeschlossenen SEPA-Formats) wie z. B. die <IBAN> aus der vor der pain message stehenden Struktur *IntlAcctInfo* visualisiert werden.

III.2.5 Spezielle Festlegungen für die Dialoginitialisierung beim Secoder-Sicherheitsverfahren

Im Rahmen der Initialisierung werden folgende Informationen ausgetauscht:

Zugelassene Secoder-Sicherheitsverfahren für den Benutzer

In der Initialisierungsantwort wird dem Kunden im Rahmen der Rückmeldungen zum Auftragsteil über den Rückmeldungscode 3921 und entsprechende Rückmeldungsparameter mitgeteilt, welche konkreten Secoder-Sicherheitsverfahren für ihn zugelassen sind. Dabei wird pro Rückmeldeparameter ein Verfahrenskennzeichen übermittelt. Derzeit ist nur der Wert „811“ möglich.

Das Kreditinstitut muss organisatorisch sicherstellen, dass der Kunde über eine geeignete Version eines Kundenproduktes verfügt, das die Rückmeldeparameter entsprechend interpretieren kann. In jedem Falle sollte der Kunde durch einen verständlichen Rückmeldungstext darauf hingewiesen werden, dass er ggf. ein aktualisiertes Kundenprodukt benötigt.

Sollte der Kunde vertraglich an die Nutzung eines der Secoder-Sicherheitsverfahren gebunden sein und verwendet er ein Kundenprodukt, welches dieses Secoder-Sicherheitsverfahren nicht unterstützt, so ist die Kommunikation zu beenden. Über den Rückmeldungscode 9955 „Secoder-Sicherheitsverfahren nicht zugelassen“ und einen geeigneten Rückmeldungstext muss der Kunde eindeutig über die Ursache dieser Beendigung der Kommunikation informiert werden. Der Rückmeldungstext muss auch berücksichtigen, dass die Anfrage des Kundenproduktes z. B. mit *Option* = „999“ in diesem Fall nur erfolgt, um die unterstützten Sicherheits-Verfahren für den Benutzer zu ermitteln. Diese müssen über den Rückmeldungscode 3921 „Zugelassene Secoder-Sicherheitsverfahren für den Benutzer“ (oder den entsprechenden Rückmeldungscode 3921 in Kombination mit Code 9800 im Fehlerfall Dialogabbruchfall mit *TermSession*) mitgeteilt werden. Beim Secoder-Sicherheitsverfahren=811 gilt dies ab der ersten Kommunikation nach der Schlüsseleinreichung (mit *Option* = 7 oder 9).

Sollte das Kundenprodukt Secoder-Sicherheitsverfahren unterstützen und noch keine Verfahrensparameter mit Angabe der für den aktuellen Benutzer unterstützten Verfahren verfügen, so muss es eine Kommunikation eröffnen, um über die Rückmeldeparameter in Kenntnis der erlaubten Verfahren zu gelangen. Hierbei ist für das Element *Option* der Wert „999“ für Ein-Schritt-TAN-Verfahren zu verwenden.

Gewähltes Secoder-Sicherheitsverfahren des Kunden

Ein Kunde kann aus den für ihn zugelassenen Secoder-Sicherheitsverfahren eines für den aktiven Dialog auswählen. Das entsprechende Kennzeichen wird in das Element „Secoder Signaturverfahren“ (Tagname: *Signature*) in der Struktur *SigChoiceReq* der Initialisierungsnachricht eingestellt. Als Kodierung ist derzeit nur der Wert „811“ möglich. Das gewählte Secoder-Sicherheitsverfahren muss für den Benutzer erlaubt sein (BPD und UPD, Rückmeldung 3921 bei Dialoginitialisierung).

III.3 Dialogspezifikation für Secoder-Sicherheitsverfahren

III.3.1 Allgemeines

Alle für den Einsatz von Secoder-Sicherheitsverfahren erforderlichen Informationen werden in der Struktur *SigChoicReq*, *Secoder-Signature* transportiert. Dies bezieht sich auf sämtliche Signatur- und Visualisierungsdaten, unabhängig davon, ob eine Visualisierung am Secoder erfolgt oder nicht.

III.3.1.1 Verschlüsselung des Dialoges

Grundsätzlich sind bei Secoder-Sicherheitsverfahren sowohl alle Kunden- als auch alle Kreditinstitutsnachrichten eines Dialoges mit HBCI zu verschlüsseln. Von dieser Regel ausgenommen sind die folgenden Dialogarten:

- Anonymer Zugang
- Schlüsselsperrung durch den Kunden
- Kommunikationszugang anfordern

Im Unterschied zu der standardmäßigen Belegung in FinTS wird bei Secoder-Sicherheitsverfahren im Datenelement *Option* der Wert für das Secoder-Sicherheitsverfahren, also 811 eingestellt. Hierdurch wird indirekt auch das eigentliche Verschlüsselungsverfahren „HBCI“ festgelegt.

Die sonstigen Protokolleigenschaften zum Verschlüsseln von Daten sind den entsprechenden Vorgaben des Sicherheitsverfahrens HBCI (HBCI-Verschlüsselung) zu entnehmen.

III.3.1.2 Institutssignaturen bei Secoder-Sicherheitsverfahren

Beim Secoder-Sicherheitsverfahren 811 werden keine Institutssignaturen eingesetzt.

III.3.1.3 Key-Management bei Secoder-Sicherheitsverfahren

Beim Secoder-Sicherheitsverfahren 811 wird davon ausgegangen, dass das Key- bzw. Zertifikatsmanagement mit anderen syntaktischen Mitteln erfolgt.

Daher muss die erstmalige Schlüsseleinreichung nach dem Standard RAH-Verfahren erfolgen. Zum Einreichen der Schlüssel muss also zunächst eine separate Kommunikation mit *Option* = 7 bzw. 9 durchgeführt werden. Gleiches gilt für die Schlüsselsperre. Änderungen der Kundenschlüssel könnten bei den kartenbasierten Secoder-Sicherheitsverfahren nicht auftreten.

Eine Ausnahme stellt das Anfordern der öffentlichen Schlüssel des Instituts dar. Der Austausch erfolgt über die entsprechenden Segmente im Rahmen des jeweiligen Secoder-Sicherheitsverfahrens.

IV. CHIPAPPLIKATIONEN

IV.1 Chipapplikation für RAH

Kapitel *IV.1.1 Applikation Notepad* spezifiziert die Datenstrukturen und Zugriffsregeln der Chipapplikation ”DF_NOTEPAD” für SECCOS-Chipkarten [DF_NOTEPAD]. Kapitel *IV.1.3 Terminalabläufe* spezifiziert die Terminalabläufe im Umgang mit dem RAH-Verfahren auf SECCOS-Chipkarten [SECCOS-6].

Im Verlauf dieses Kapitels ist mit ”Bankensignaturkarte” eine Chipkarte mit SECCOS-Betriebssystem und Signaturanwendung gemeint, die u. U. auch die Notepad-Applikation aus *IV.1.1 Applikation Notepad* enthält. Weitere Applikationen, wie z. B. die elektronische Geldbörse, sind nicht notwendigerweise auf der Chipkarte enthalten. Ebenso kann die Karte kontobezogen oder kontoungebunden sein. Ebenso kann die Bankensignaturkarte mit oder ohne Zertifikat ausgeliefert werden.

IV.1.1 Applikation Notepad

Die Anwendung „Notepad“ (vgl. [DF_NOTEPAD]) dient als „Notizbuch“ zur Aufnahme von Daten anderer Anwendungen. Durch das Notizbuch wird somit ein mobiler Datenspeicher geschaffen, in dem bestimmte anwendungs- bzw. benutzerspezifische Parameter abgelegt werden können, z. B. für die Bankverbindungsdaten in FinTS oder EBICS.

Wenn eine Anwendung auf die Karte zugreift, wird geprüft, ob auf der Chipkarte das Notizbuch DF_NOTEPAD vorhanden ist. Falls ja, werden die Daten ausgelesen, falls nein, muss der Benutzer die Zugangsdaten selbst eingeben bzw. die Zugangsdaten werden im Kundensystem selber verwaltet.

Im Datenspeicher EF_NOTEPAD kann jeder Record durch eine Anwendung belegt werden. Die Unterscheidung der Zugehörigkeit bestimmter Dateninhalte erfolgt an Hand der Tags eines Records:

- ’00‘ bedeutet, dass der Record nicht belegt ist
- ’F0‘ bedeutet, dass der Record HBCI-Bankverbindungsdaten (HBCI-Parameterblock) enthält.
- ’F1‘ bedeutet, dass der Record Bankverbindungsdaten analog dem DFÜ-Abkommen enthält.

Weitere Kennungen sind für den späteren Gebrauch durch andere Anwendungen vorgesehen (Tag ‘F2‘ bis ‘FE‘).

Somit können mehrere FinTS-Bankverbindungsdaten (im Sinne der Multibankfähigkeit) in unterschiedlichen Records, jeweils mit Kennung/Tag ’F0‘ abgelegt werden. Jede FinTS-Bankverbindung belegt dabei einen Record.

IV.1.2 EF_NOTEPAD

Bei dem EF_NOTEPAD handelt es sich um ein lineares EF mit einer variablen Recordlänge, die aus technischen Gründen auf maximal 239[1] Byte begrenzt ist. Es dient der Ablage beliebiger Daten.

Die FinTS Anwendung nutzt das EF_NOTEPAD zur Speicherung von zugangsspezifischen Daten, den HBCI-Parameterblöcken. So kann ein Online-Banking-Kundenprodukt in einem HBCI-Parameterblock und damit in einem Record des EF_NOTEPADs Informationen wie z. B. die FinTS-Benutzerkennung ablegen. Darüber hinaus können vom Kundenprodukt in einem separaten weiteren Record aber auch (produktspezifische) Informationen zu Kundenpräferenzen und –einstellungen (z.B. Sprache, Anzeigeparameter etc.) abgelegt werden.

Den Herstellern von Kundensystemen wird vorgeschlagen, beim EF_NOTEPAD neben einer Länge von 239 Byte auch Karten mit einer Maximallänge von nur 200 Byte zu unterstützen. Zur Ermittlung der Maximallänge soll der Tag „82“ des Bereiches FCP ausgelesen werden.

Der Inhalt des Notepad kann im Wesentlichen nur nach vorhergehender, erfolgreicher CSA-Passwort-Verifizierung gelesen und verändert werden. Somit ist der Inhalt insbesondere vor unberechtigtem Auslesen geschützt (z.B. wenn die Kontonummer als Bestandteil der Benutzerkennung gespeichert ist).

Das Auslesen der Records erfolgt über ein *Read Record* auf alle vorhandenen Records. Wird ein HBCI-Parameterblock gesucht so ist anschließend ein Vergleich durchzuführen, ob der TAG des Records den Inhalt ’F0‘ enthält.

Alternativ können mit dem Kommando SEARCH RECORD mit dem Suchmuster ’F0‘ für das erste Byte des Recordinhalts genau die für HBCI relevanten Records ausgelesen werden.

*** FCP**

Für das EF_NOTEPAD sind die folgenden FCP festzulegen:

Tag	Länge	Wert	Erläuterung
'62'	'1C'		Tag und Länge für FCP
'82'	'05'	'14 41 00 EF XX'	Datei-Deskriptor für lineares EF mit variabler Recordlänge bis zu 239 ('EF') Byte und XX Records
'83'	'02'	'A6 11'	Datei-ID des EF_NOTEPAD
'85'	'02'	'YY YY'	für Nutzdaten allokierter Speicherplatz in Byte (XX Records mal 239 Byte)[2]
'88'	'01'	'D0'	SFI '1A' für das EF_NOTEPAD
'A1'	'08'	'8B 06 00 30 01 04 02 05'	Zugriffsregel-Referenzen

[1] Nach ISO 7816-4 ist eine APDU maximal 255 Bytes lang. Nach Abzug der Protokolldaten steht eine netto Datenlänge von maximal 239 Byte zur Verfügung.

[2] Beispiel: für XX = ‘05‘ a 239 Byte ist ein Datenbereich von 1195 Byte anzulegen →YY YY = ’04 AB‘.

Die maximale Anzahl der Records und deren maximale Länge werden bei der Produktion der Karte festgelegt.

Im SE #1 dürfen READ, SEARCH und UPDATE RECORD nur ausgeführt werden, wenn zuvor eine Karteninhaberauthentikation mit dem globalen Passwort 3 (CSA-Passwort) erfolgt ist. Der Returncode wird nicht MAC-gesichert (Zugriffsregeln im Record 4 des EF_RULE).

Im SE #2 dürfen die Kommandos READ, SEARCH und UPDATE RECORD nur ausgeführt werden, wenn sie mit Secure Messaging durchgeführt werden. **Entweder** ist zuvor eine Karteninhaberauthentikation mit dem globalen Passwort 3 (CSA-Passwort) erfolgt und die MAC-Bildung im Secure Messaging erfolgt für Kommando- und Antwortnachricht mit dem Sessionkey SK2; **oder** (ohne vorherige Karteninhaberauthentikation) die MAC-Bildung erfolgt für Kommando- und Antwortnachricht mit dem $K_{Notepad\_Admin}$ (Zugriffsregel im Record 5 des EF_RULE).

Im SE #2 darf das Kommando APPEND RECORD nur mit Secure Messaging durchgeführt werden. Die MAC-Bildung erfolgt für Kommando- und Antwortnachricht mit dem $K_{Notepad\_Admin}$.

Im SE #2 darf das Kommando SELECT FILE (EF) ohne Einhaltung von Zugriffsbedingungen oder mit Secure Messaging durchgeführt werden. Die MAC-Bildung im Secure Messaging erfolgt für Kommando- und Antwortnachricht mit dem Sessionkey SK2.

*** Aufbau eines Records**

POS	Länge	Wert	Erläuterung
1	1	'XX'	Tag
2	1 oder 2	'XX' oder '81 XX'	Länge (bei Längen über 127 Byte ist die Kodierung '81' 'xx' zu verwenden)
3	L	'XX..XX'	Nutzdaten

Als Tags werden festgelegt:

Byte 1	Bedeutung
'00'	freier Record
'F0'	Belegung mit HBCI-Parameterblock
'F1'-'FE'	RFU

Durch den Tag 'F0' wird ein Recordeintrag als HBCI-Parameterblock für die HBCI-Anwendung gekennzeichnet. Für Belegungen der EF_NOTEPAD-Records durch andere Anwendungen stehen die Tags 'F1' bis 'FE' zur Verfügung. Die Kennungen werden durch die Deutsche Kreditwirtschaft vergeben.

Initial werden alle Records mit '00..00' belegt und so als leere Records gekennzeichnet.

*** Beispiel eines EF_NOTEPADs**

In der folgenden Tabelle ist die beispielhafte Belegung eines EF_NOTEPAD mit 7 Records angegeben.

Record	Eintrag	Erläuterung
1	’F0 XX...XX‘	Erste HBCI-Bankverbindung
2	’F0 XX...XX‘	Zweite HBCI-Bankverbindung
3	’F0 XX...XX‘	Dritte HBCI-Bankverbindung
4	’00..00‘	frei
5	’F1 XX..XX‘	belegt durch Anwendung mit Kennung ’F1‘
6	’00..00‘	frei
7	’F0 XX...XX‘	Vierte HBCI-Bankverbindung

*** Umgang mit variablen Recordlängen**

Durch die Definition des EF_NOTEPAD als lineares EF mit variabler Recordlänge werden beim Lesen eines Records nur die tatsächlich vorhandenen Daten von der Karte zurückgegeben.

Command APDU eines READ RECORD:

Byte	Wert	Erläuterung
1-2	'00 B2'	CLA, INS
3	'0X'	P1, Recordnummer X
4	'D4'	P2, Reference Control Byte
5	'00'	L_e

Wenn das READ RECORD erfolgreich ausgeführt wird, gibt die Chipkarte eine Antwortnachricht mit der folgenden Struktur zurück:

Byte	Länge	Wert	Erläuterung
1-L	L	'XX ...XX'	Recordeintrag
(L+1)-(L+2)	2	‘SW1 SW2‘	Positiver Returncode SW1 SW2

Ein HBCI-Recordeintrag beginnt in diesem Fall mit dem Tag ‘F0‘ und einem Längenbyte.

IV.1.2.1 Recordbelegung des EF_NOTEPAD mit einem HBCI-Parameterblock

Ein HBCI-Recordeintrag hat folgenden prinzipiellen Aufbau:

Tag		Länge (Byte)	Wert	For-mat	Sta tus	Erläuterung
'F0'		Var. max 'EC'[3]				HBCI-Parameterblock
	'C0'	'03'	'30' '30' '32'	3an	M	Version 002 des HBCI-Parameterblocks
'E1'		Var. max. '5B'			M	HBCI-Institutsparameterblock
	'C1'	'01'-'14'	Kreditinstituts-bezeichnung	..20an	O	
	'C2'	'03'	Länderkenn-zeichen	3an	M	ISO 3166 numerisch in 3 ASCII-Zeichen codiert
	'C3'	'01'-'1E'	Kreditinstitutscode	..30an	M	in jeweils national bekannter Notation
	'C4'	'27'	Hashwert Instituts-schlüssel	39bin	O	
	'C5'	'01'	Schlüsselstatus	1bin	M	8 Statusflags
'E2'		Var. max. '37'			M	HBCI-Kommunikations-parameterblock
	'C6'	'01'	Kommunikations-dienst	1n	M	2 = TCP/IP
	'C7'	'01'-'32'	Kommunikations-adresse	..50an	M	
'E2'		Var. max. '37'			O	2. HBCI-Kommunikations-parameterblock
	'C6'	'01'	Kommunikations-dienst	1n	M	2 = TCP/IP
	'C7'	'01'-'32'	Kommunikations-adresse	..50an	M	
'E3'		Var. max. '54'			O	HBCI-Kundenparameterblock
	'C8'	'01'-'1E'	Benutzerkennung	..30an	M	
	'C9'	'01'-'1E'	Kunden-ID	..30an	O	
	'CA'	'0C' oder '12''	Info Inhaber-schlüssel	12an oder 18an	M	Schlüsselnummer und Schlüs-selversion jeweils für den Sig-nierschlüssel, den Chiffrier-schlüssel und optional für den Signaturschlüssel des Karten-inhabers

IV.1.2.1.1 Tag 'F0': HBCI-Parameterblock

Durch das Tag 'F0' wird ein Record mit HBCI-Parameterblock für die HBCI-Anwendung gekennzeichnet. Für Belegungen der EF_NOTEPAD-Records durch andere Anwendungen stehen die Tags 'F1' bis 'FE' zur Verfügung.

[3] Nettodatenlänge ‚EC'=236 Byte + 3 Byte Längenfeld ergibt die maximale Recordlänge von 239 Byte

Ein HBCI-Parameterblock enthält in der angegebenen Reihenfolge:

- **optional** ein Versionskennzeichen
- genau einen HBCI-Institutsparameterblock mit **Tag 'E1'**
- genau einen HBCI-Kommunikationsparameterblöcke mit **Tag 'E2'**
- **optional** einen weiteren HBCI-Kommunikationsparameterblöcke mit **Tag 'E2**[4]
- **optional** einen HBCI-Kundenparameterblock mit **Tag 'E3'**

Die maximale Länge des HBCI-Parameterblocks wird beschränkt durch die maximale Recordlänge von 239 Byte[5].

IV.1.2.1.2 Tag 'C0': HBCI-Version

In jedem 'F0' Record kann zur Kennzeichnung der Version des EF-NOTEPAD ein Sub-Record (z. B 'C0' '03' '30' '30' '30') aufgenommen werden. Die Zählung der Version beginnt bei 1. Ist kein Sub-Record 'C0' vorhanden, so bedeutet dieses, dass die Belegung des EF-NOTEPAD gemäß der Version 1 erfolgt. Diese Recordbelegung ist jedoch ab FinTS V4.1 nicht mehr zugelassen.

IV.1.2.1.3 Tag 'E1': HBCI-Institutsparameterblock

Durch das Tag 'E1' wird der Block der institutsspezifischen Parameter gekennzeichnet. Ein HBCI-Institutsparameterblock enthält in der angegebenen Reihenfolge:

- **optional** eine Kreditinstitutsbezeichnung mit **Tag 'C1'**, alphanumerisch mit bis zu 20 Zeichen
- genau ein Länderkennzeichen des kontoführenden Instituts mit **Tag 'C2'**. Verwendet wird der numerische ISO 3166-Code als 3-stellige alphanumerische Zeichenkette (z.B. Deutschland = "280")
- genau eine Kreditinstitutskennung mit **Tag 'C3'**, in einer jeweils national bekannten Notation mit bis zu 30 Stellen. Für deutsche Kreditinstitute wird hier die 8-stellige Bankleitzahl (gemäß FinTS Datenelement ‚Kreditinstitutscode') verwendet.

4 Somit ist der erste HBCI-Kommunikationsparameterblock ist also verpflichtend, der zweite optional.

5 In einer konkreten Umsetzung ist es nicht möglich einen HBCI-Parameterblock mit allen Felder in der maximalen Länge zu nutzen. Dabei würde die maximale Recordlänge von 239 Byte überschritten.

- **optional** einen Hashwert des öffentlichen Signierschlüssels des Instituts mit **Tag 'C4'**, binär mit genau 39 Byte für das Hashwertverfahren SHA-256. Das Verfahren ist abhängig vom Sicherheitsprofil zu wählen.
  - Der Eintrag besteht bei SHA-256 aus

    [3 Byte Schlüsselnummer | 3 Byte Schlüsselversion | 1 Byte Kennzeichen Hashverfahren | 32 Byte Hashwert].

    Als Kennzeichen für das Hashverfahren wird festgelegt:

    - '03' = SHA-256 für RAH-7 und RAH-9

    Die Parameter Schlüsselnummer und Schlüsselversion des Institutsschlüssels werden in je 3 Byte rechtsbündig mit führenden Nullen codiert (z.B. Schlüsselnummer 1 → die Bytefolge '30' '30' '31').

- genau ein Schlüsselstatus mit **Tag 'C5'**, binär von genau 1 Byte Länge. Der Schlüsselstatus enthält acht Flags mit folgender Bedeutung:

Bit1	Erstmalige Übermittlung der Kundenschlüssel notwendig	'1'b - Ja '0'b - Nein
Bit2	Institutsrechner erwartet Signaturen nach ISO9796 mit AnnexA	'1'b - Ja '0'b - Nein
Bit3	Institutsschlüssel validiert	'1'b - Ja '0'b - Nein
Bit4	Ausstehende Übermittlung des neuen öffentlichen Chiffrierschlüssels des Kunden bei Schlüsseländerung[6]	'1'b - Ja '0'b - Nein
Bit5	Ausstehende Übermittlung des neuen öffentlichen Signierschlüssels des Kunden bei Schlüsseländerung[7]	'1'b - Ja '0'b - Nein
Bit6	Schlüsselsperre mit Erfolg durchgeführt (Info, da terminierte Sperrung erst in der Zukunft wirksam werden kann)	'1'b - Ja '0'b - Nein
Bit7	Leitungsprobleme bei Übermittlung neuer Schlüssel	'1'b - Ja '0'b - Nein
Bit8	Reserviert	'0'b

Bei der Personalisierung muss als Initialisierungswert '01' aufgebracht werden.

Ein HBCI-Institutsparameterblock belegt inklusive der Tag- und Längenbytes somit maximal 93 Byte.

[6] Nicht zu belegen, da die DK-Bankensignaturkarte keinen Wechsel der Kundenschlüssel unterstützt.

[7] Nicht zu belegen, da die DK-Bankensignaturkarte keinen Wechsel der Kundenschlüssel unterstützt.

IV.1.2.1.4 Tag ‘E2‘: HBCI-Kommunikationsparameterblock

Durch das Tag ‘E2‘ wird der Block der generellen Kommunikations-Parameter gekennzeichnet. Ein HBCI-Kommunikationsparameterblock enthält in der angegebenen Reihenfolge:

- genau einen Kommunikationsdienst mit **Tag ‘C6‘**, 1 Stelle numerisch. Zurzeit definiert ist der numerische Wert 2 (TCP/IP)
- genau eine Kommunikationsadresse mit **Tag ‘C7‘**, alphanumerisch mit bis zu 50 Zeichen

Ein HBCI-Kommunikationsparameterblock belegt inklusive der Tag- und Längenbytes somit maximal 57 Byte.

IV.1.2.1.5 Tag ‘E3‘: HBCI-Kundenparameterblock

Durch den Tag ‘E3‘ wird der **optional** vorhandene Block der kundenspezifischen Parameter gekennzeichnet. Ist der Block nicht vorhanden, so handelt es sich um eine im Rahmen der HBCI-Anwendung Bankensignaturkarte ohne Zertifikat. Ein HBCI-Kundenparameterblock enthält in der angegebenen Reihenfolge:

- genau eine Benutzerkennung mit **Tag ‘C8‘**, alphanumerisch mit bis zu 30 Zeichen
- **optional** eine Kunden-ID mit **Tag ‘C9‘**, alphanumerisch mit bis zu 30 Zeichen
- genau ein Info Inhaberschlüssel mit Tag 'CA', von genau 12 oder 18 numerischen Zeichen**.**
- Bei 12 Byte Länge des Blocks ist der Inhalt wie folgt definiert:

Schlüsselnummer Signierschlüssel [3n]
Schlüsselversion Signierschlüssel [3n]
Schlüsselnummer Chiffrierschlüssel [3n]
Schlüsselversion Chiffrierschlüssel [3n]

- Bei 18 Byte Lange ist der Inhalt wie folgt definiert:

Schlüsselnummer Signierschlüssel [3n]
Schlüsselversion Signierschlüssel [3n]
Schlüsselnummer Chiffrierschlüssel [3n]
Schlüsselversion Chiffrierschlüssel [3n]
Schlüsselnummer Signaturschlüssel [3n]
Schlüsselversion Signaturschlüssel [3n]

Die Parameter Schlüsselnummer und Schlüsselversion werden in je 3 Byte numerisch rechtsbündig mit führenden Nullen angegeben. (z.B. Schlüsselnummer 1 → "001" → die Bytefolge '30' '30' '31'.

Fehlen die Angaben für den Signaturschlüssel (CA Record der Länge 12 Byte) so werden als Schlüsselnummer und Schlüsselversion des Signaturschlüssels die Schlüsselnummer und Schlüsselversion des Signierschlüssels übernommen.

Fehlt der Teilrecord mit dem Tag 'CA' (nicht vorhandener Record E3 oder Record CA oder fehlendes EF_NOTEPAD) und liegen somit weder für den Signierschlüssel und den Chiffrierschlüssel noch für den Signaturschlüssel Schlüsselnummer und Schlüsselversion vor so sind vom FinTS-Kundensystem die Schlüsselnummern und Schlüsselversionen aller Schlüssel nach folgenden Mechanismen vorzubesetzen.

Die Schlüsselnummer wird gemäß dem genutzten RAH-Verfahren besetzt. Die Schlüsselversion wird gängigerweise im ersten Ausgabejahr mit "001" vorbesetzt und anschließend im jährlichen Turnus um 1 erhöht.

RAH Verfahren	Schlüsselnummer	Schlüsselversion
RAH9	"009" → '30' '30' 39'	"001" → '30' '30' '31'
RAH10	"010" → '30' '31' 30'	"001" → '30' '30' '31'

Über die Schlüsselnummer im EF_NOTEPAD kann das zu verwendende Sicherheitsprofil ermittelt werden.

Wichtiger Hinweis:
Bei allen Verfahren müssen für die Schlüsselnummer die entsprechenden Werte aus der obigen Tabelle verwendet werden. Die Nutzung von Schlüsselnummer „001“ ist nicht erlaubt.

Ein HBCI-Kundenparameterblock belegt inklusive der Tag- und Längenbytes somit maximal 86 Byte.

IV.1.2.1.6 Beispiel

Beispiel für eine Recordbelegung (Tags und Längenbytes sind fett markiert)

Inhalt	Erläuterung
F0 81 76	HBCI-Parameterblock
E1 3D	Institutsparameterblock
C1 0C 54 45 53 54 49 4E 53 54 49 54 55 54	Institutsbezeichnung “TESTINSTITUT“
C2 03 32 38 30	Länderkennzeichen “280“
C3 08 31 32 33 34 35 36 37 38	BLZ 12345678
C4 1B 30 30 31 30 30 31 03 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20	Schlüsselnummer 1, Schlüsselversion 1, Hashverfahren SHA-256, Hashwert
C5 01 01	Schlüsselstatus '01'
E2 12	Kommunikationsparameterblock
C5 01 02	Kommunikationsdienst TCP/IP
C6 0D 31 39 32 2E 31 36 38 2E 31 31 2E 32 32	Kommunikationsadresse 192.168.11.22
E3 21	Kundenparameterblock
C8 0A 31 32 33 34 35 36 37 38 39 30	Benutzerkennung

	"1234567890"
C9 05 31 32 33 34 35	Kunden-ID "12345"
CA 0C 30 30 31 30 30 31 30 30 31 30 30 31	Info Inhaberschlüssel Schlüsselnummer SIG 1, Schlüsselversion SIG 1 Schlüsselnummer CHIF 1, Schlüsselversion CHIF 1

IV.1.2.1.7 Erreichen der maximalen Recordlänge

Bei Ausnutzung aller Maximallängen und Aufnahme aller optionalen Felder und Angabe zweier Kommunikationsparameterblöcke und eines Kundenparameterblocks ergibt sich ein maximaler Platzbedarf von 297 Byte. Dieser Platzbedarf ist aber in einem Record nicht abbildbar. Normalerweise wird aber nur ein Kommunikationsparameterblock verwendet sowie selten alle Maximallängen ausgereizt, so dass meistens die maximale Recordlänge von 239 Byte genügt. Bei älteren bereits ausgegebenen Bankensignaturkarten ist nur eine maximale Recordlänge von 200 Byte vorgesehen.

IV.1.3 Terminalabläufe

Dieses Kapitel spezifiziert die Terminalabläufe im Umgang mit dem RAH-Verfahren auf SECCOS-Chipkarten [SECCOS-6] bzw. [SECCOS-7]. Ein Online-Banking-Kundensystem nutzt

- zur Verschlüsselung und Signierung von FinTS-Nachrichten die auf der Chipkarte zur Verfügung stehende Signatur-Anwendung (DF_SIG, [ZKASIG]) und die durch das Betriebssystem bereitgestellten Signatur-Funktionen,
- als Sequenzzähler (Signatur-ID) interne Bedienungszähler der Signatur-Anwendung (siehe IV.1.3.1 Verfahren zur Ermittlung der Sicherheitsreferenznummern),
- als Datenspeicher für die Zugangsdaten ein auf der Chipkarte optional vorhandenes DF_NOTEPAD ([NOTEPAD], siehe IV.1.1 Applikation Notepad).

IV.1.3.1 Verfahren zur Ermittlung der Sicherheitsreferenznummern

Auf der Bankensignaturkarte wird kein eigenständiger Sequenzzähler verwaltet, sondern es werden jeweils chipkarteninterne „Usage Counter“ der beiden zur Signatur verwendeten Schlüssel S_K.CH.DS und S_K.CH.$AUT_{C/S}$ herangezogen.

Für jedes Signaturschlüsselpaar wird ein separater Usage Counter verwaltet. Dieser kann jeweils zwei, drei oder vier Byte lang sein.

Da die Usage Counter auf der Chipkarte dekrementiert werden, als Sicherheitsreferenznummer („Signatur-ID“) aber ein streng monoton aufsteigender Zähler gefordert ist, wird die konkrete Sicherheitsreferenznummer nach folgendem Algorithmus ermittelt:

1. Auslesen des 2 bis 4 Byte langen Usage Counter (UC) UC_{DS} des Schlüssels S_K.CH.DS bzw. UC_{AUT} des Schlüssels S_K.CH.$AUT_{C/S}$.
2. Sei **neg**(UC) die bitweise logische Negation von UC. Dann ist die Sicherheitsreferenznummer (SRN)

$$SRN_{DS} = \mathbf{neg}(UC_{DS})$$

$$SRN_{AUT} = \mathbf{neg}(UC_{AUT})$$

Die einzelnen Usage Counter haben folgende Wertebereiche:

von 0 bis 65.535	bei Länge(UC) = 2 Byte
von 0 bis 16.777.215	bei Länge(UC) = 3 Byte
von 0 bis 4.294.967.295	bei Länge(UC) = 4 Byte

Damit muss die Sicherheitsreferenznummer SRN über die entsprechenden Wertebereiche verfügen und benötigt zur Darstellung ebenfalls mindestens 2, 3 oder 4 Byte.

Ein Wrap-Around bei Erreichen des jeweiligen Maximalwerts findet nicht statt, da das Erreichen eines Usage Counter 0 den Schlüssel der Chipkarte für die weitere Verwendung sperrt.

Beispiel:

UC_{DS} = ’00 0A‘ (dezimal 10) $\Rightarrow$ SRN_{DS} = **neg**(UC_{DS}) = ‘FF F5‘ (dezimal 65.525)

UC_{AUT} = ‘FA 1D‘ (dezimal 64.029) $\Rightarrow$ SRN_{AUT} = **neg**(UC_{AUT}) = ’05 E2‘(dezimal 1506)

Dieser Algorithmus ist in der jeweiligen Anwendungssoftware zu realisieren.

IV.1.3.2 Beschreibung der Terminalabläufe

Nachfolgend werden die Anwendungsabläufe aus Endgerätesicht an einem privaten Signaturterminal [KT-KONZEPT] spezifiziert. Hierbei werden ausschließlich die chipkartenbezogenen Aspekte berücksichtigt. Anwendungsbezogene Details sind nicht Bestandteil dieser Spezifikation.

Um die Abläufe möglichst einfach beschreiben zu können, werden in der nachfolgenden Beschreibung Befehle der ZKA-SIG-API [KT-SIG] verwendet. Hiermit ist jedoch die Verwendung der ZKA-SIG-API für technische Implementierungen nicht zwingend vorgeschrieben. Wird die ZKA-SIG-API nicht verwendet, so sind die in [KT-SIG] angegebenen Abläufe zum Aufruf der KT-Kommandos zu berücksichtigen.

Die Anwendungsabläufe lassen sich auch auf öffentliche Signaturterminals (Geschäftsterminals) erweitern. Zu beachten ist dabei insbesondere, dass in diesem Fall zusätzlich eine

- Komponenten-Authentifikation zwischen Chipkarte und Geschäftsterminal mit Aushandlung eines Session-Key-Paares (SK1, SK2) stattfindet;
- alle Befehle an die Chipkarte im Secure Messaging mit einem SK2-MAC durchgeführt werden müssen.

Falls bei der Ausführung der Kommandos ein Fehler auftritt, bricht das Terminal den Vorgang ab, es sei denn, es ist ein abweichendes Verhalten spezifiziert.

In den hier beschriebenen Abläufen ist das Kundenterminal durch ein *zka_sig_open* (zu Beginn des Ablaufs „Signatur einleiten“) und ein *zka_sig_close* (Am Ende des Ablaufs „Signatur beenden“) für die gesamte Zeitdauer exklusiv für die FinTS-Kundenanwendung reserviert.

Um zwischenzeitlich anderen Anwendungen die Möglichkeit zu geben, die Signaturdienste der Karte zu nutzen (z. B. für die Zeitdauer der Nachrichtengenerierung), können die im Folgenden beschriebenen Teilabläufe jeweils auch durch ein *zka_sig_open* und ein *zka_sig_close* gekapselt werden. Dadurch wird die exklusive Reservierung des Kundenterminals aufgehoben, die internen Zwischenwerte der ZKA-SIG-API (insbes. der Chipdaten) bleiben jedoch erhalten. Erst durch Aufruf des *zka_sig_fini_signature_application* im Ablauf „Signatur beenden“ werden die internen Zwischenwerte der ZKA-SIG-API gelöscht.

Zur Administration der Signaturkarten (z. B. Freischalten eines Zertifikates, Rücksetzen des Fehlbedienungszählers) werden von den Kreditinstituten bzw. den Kartenemittenten Softwarekomponenten zur Verfügung gestellt werden, die in der privaten Kundenumgebung zum Einsatz kommen sollen. In Homebanking-Kundensystemen, die nicht von den Kartenemittenten herausgegeben werden, sollen diese Administrationsfunktionen nicht realisiert werden.

Für die kreditinstitutsseitige Realisierung dieser Softwarekomponenten hat Die Deutsche Kreditwirtschaft Anforderungen und Festlegungen formuliert, die bei Bedarf über die jeweiligen Ansprechpartner der Standards erhältlich sind.

IV.1.3.2.1 Signatur einleiten

Chipkarte			Endgerät	
			M1	Aufruf der ZKA-SIG-API-Funktion *zka_sig_open*
		←	M2	Aufruf der ZKA-SIG-API-Funktion *zka_sig_init_signature_application*
R2	OK	→		
		←	M3	Aufruf der ZKA-SIG-API-Funktion *zka_sig_verify_CSA_password*
R3	OK	→		
		←	C4	SELECT FILE DF_NOTEPAD
R4	OK / „File not found“	→		
		←	C5	ggf. READ RECORD EF_NOTEPAD
R5	Bankverbindung	→		
			A5	Daten prüfen und speichern

- **Erläuterung**

1. Die ZKA-SIG-API-Funktion *zka_sig_open* wird ausgeführt. Diese Funktion stellt eine exklusive Verbindung zum Kundenterminal her.
2. Die ZKA-SIG-API-Funktion *zka_sig_init_signature_application* wird ausgeführt. Diese sorgt insbesondere für ein Reset der Karte und das Auslesen der relevanten Basisinformationen der Karte.
3. Die ZKA-SIG-API-Funktion *zka_sig_verify_CSA_password* wird ausgeführt. Diese Funktion liest das CSA-Passwort ein und führt eine Verifikation gegenüber der Chipkarte durch.
4. Die Applikation „Notepad“ wird geöffnet, indem das ADF der Applikation, DF_NOTEPAD durch das Terminal mittels des Kommandos SELECT FILE ausgewählt wird.

- **Command APDU**

Byte	Wert	Erläuterung
1-2	'00 A4'	CLA, INS
3	'04'	P1, Selektion mit DF-Name
4	'0C'	P2, Keine Antwortdaten
5	'09'	L_C
6-14	'D2 76 00 00 25 4E 50 01 00'	AID der Notepad-Applikation

Wenn die Notepad-Applikation auf der Karte nicht vorhanden ist, wird der folgende Schritt übersprungen. In diesem Fall müssen die Zugangsdaten von einer anderen Stelle gelesen oder vom Benutzer eingegeben werden.

4. Das Terminal liest mittels READ RECORD sukzessive die Bankverbindungsdaten in den Records des EF_NOTEPAD (SFI '1A'), bis der oder die ”passenden” Einträge gefunden wurden. Das Lesen von Einträgen ist erst nach erfolgreicher CSA-Passwort-Verifikation (Schritt 2) möglich.

♦ **Command APDU**

Byte	Wert	Erläuterung
1-2	'00 B2'	CLA, INS
3	'0X'	P1, Recordnummer X
4	'D4'	P2, Reference Control Byte
5	'00'	L_e

Wenn das READ RECORD erfolgreich ausgeführt wird, gibt die Chipkarte eine Antwortnachricht mit der folgenden Struktur zurück:

Byte	Länge	Wert	Erläuterung
1-2	2	'XX LL'	Kennung und Länge
3-LL	LL	'XX..XX'	Nutzdaten
(LL+1)-(LL+2)	2	'XX XX'	Positiver Returncode SW1 SW2

Ist die Kennung ’F0‘, so sind FinTS-Zugangsdaten gemäß *IV.1.1 Applikation Notepad* enthalten. Es werden alle weiteren Records gelesen, bis die Chipkarte das Ende der Datei (keine weiteren Records) signalisiert.

Anstatt alle Records auszulesen und auf Übereinstimmung mit der Kennung zu überprüfen, kann alternativ auch das Kommando SEARCH RECORD verwendet werden, um mittels eines übergebenen Suchmusters vorab die ”passenden” Recordnummern in einem Schritt zu finden. Anschließend müssen dann nur diese Recordnummern mittels READ RECORD ausgelesen werden.

♦ **Command APDU**

Byte	Wert	Erläuterung
1-2	'00 A2'	CLA, INS für SEARCH RECORD
3	'01'	P1, Start mit Recordnummer 1
4	'D7'	P2, spezifische Suche im SFI ‘1A‘
5	'04'	L_C
6	'04'	CTRLB
7	'00'	Offset Indicator Byte
8	'02'	Konfigurationsbyte
9	’F0‘	Suchmuster
10	'00'	L_e

Wenn das SEARCH RECORD erfolgreich ausgeführt wird, gibt die Chipkarte eine Antwortnachricht mit der folgenden Struktur zurück:

Byte	Länge	Wert	Erläuterung
1-n	n	'XX XX'	Recordnummer(n)
n+1	1	'XX'	Statusbyte SW1
n+2	1	'XX'	Statusbyte SW2

Es können nun gezielt nur die in der Antwortnachricht angegebenen Records ausgelesen werden.

IV.1.3.2.2 Nachrichten generieren

Dieser Teil des Gesamtablaufs ist nur insofern chipkartenrelevant, als (optional) Bankverbindungsdaten, die für die Auftragsgenerierung benötigt werden, aus der Chipkarte entnommen werden. Dies ist bereits im Schritt „Signatur einleiten“ (*IV.1.3.2.1 Signatur einleiten*) geschehen. Für die folgende Ablaufbeschreibung wird angenommen, dass die Anwendung bereits FinTS-Nachrichten generiert hat. Diese Nachrichten müssen jetzt ggf. noch kryptographisch gesichert werden, d. h. es werden Segmente für die elektronische(n) Signatur(en) und für die Verschlüsselung entsprechend den FinTS-Spezifikationen eingefügt.

IV.1.3.2.3 Nachrichten signieren

Die folgenden Abläufe können offline, d. h. außerhalb eines FinTS-Dialoges vollzogen werden. Dies gilt nicht für die Erstellung von Botensignaturen. Der Grund besteht darin, dass für die Absicherung aller Kreditinstitutsnachrichten der Schlüssel des Boten erforderlich ist. Daher muss während eines gesamten Dialoges die Chipkarte des Boten im Endgerät stecken.

Die Abläufe für die Botensignatur sind grundsätzlich identisch mit den im Folgenden beschriebenen Abläufen für die Erstellung von Auftragssignaturen. Da aber ggf. für die Botensignatur anwendungsseitig noch weitere Chipkartendaten (Benutzerkennung, Benutzerreferenz, Kommunikationszugang etc.) benötigt werden, wird der komplette Ablauf in *IV.1.3.2.5 FinTS-Dialog führen* noch einmal beschrieben.

Chipkarte			Endgerät	
R1	BZ	→ ←	M1	Sequenzzähler (Signatur-ID) ermitteln durch Aufruf der ZKA-SIG-API-Funktion *zka_sig_read_key_usage_counter* und anschließende Invertierung des Rückgabewerts gemäß Abschnitt IV.1.3.1
			A2	Signatur aufbauen und in FinTS-Nachricht einfügen
			A3	Daten (Signatur-Segment, FinTS-Nutzdaten) für Signaturerstellung bereitstellen
		→ ←	M4	Signaturerstellung (siehe *IV.1.3.3.1 Signatur-Berechnung*)
			A5	ggf. M1 bis M4 für weitere Nachrichten wiederholen
			A6	signierte FinTS-Nachrichten zur Weiterverarbeitung speichern

♦ **Erläuterung**

1. Der Sequenzzähler (Signatur-ID) wird durch Auslesen der Bedienungszähler der Signaturanwendung und anschließende Berechnung ermittelt. Das Auslesen erfolgt durch Aufruf der ZKA-SIG-API-Funktion *zka_sig_read_key_usage_counter* mit der Parameterbelegung

   - counter_type = ’00‘ bei Verwendung des S_K.CH.DS, bzw.
   - counter_type = ‘02‘ bei Verwendung des S_K.CH.$AUT_{C/S}$

   Das Ergebnis BZ wird gemäß IV.1.3.1 zu SZ = **neg**(BZ) invertiert und als Sequenzzähler gespeichert.

2. Das Signatur-Segment wird aufgebaut und in die FinTS-Nachricht eingefügt.
3. Die Daten (Signatur-Segment, FinTS-Nutzdaten) für die Signaturerstellung werden bereitgestellt.
4. Die Signatur wird berechnet (siehe hierzu *IV.1.3.3.1 Signatur-Berechnung*).

5. Ggf. können die Schritte 1 bis 4 für weitere Nachrichten wiederholt werden.
6. Die signierten FinTS-Nachrichten können zur Weiterverarbeitung gespeichert werden.

Anmerkung: Für Mehrfachsignaturen wird jeweils die Abfolge „Signatur einleiten“ – „Nachrichten signieren“ – „Signatur beenden“ wiederholt. Dies kann auch zu einem späteren Zeitpunkt geschehen. Mehrfachsignaturen müssen jedoch abgeschlossen sein, bevor die Verschlüsselung der Nachricht (*IV.1.3.2.4 Nachrichten verschlüsseln*) durchgeführt wird.

IV.1.3.2.4 Nachrichten verschlüsseln bei HBCI

Die Chipkarte ist bei der eigentlichen Nachrichtenverschlüsselung nicht involviert. Die Software berechnet einen Einmalschlüssel, verschlüsselt das Dokument und verschlüsselt den Einmalschlüssel zur Übertragung mit dem öffentlichen Key-Encryption-Schlüssel $P_K.RECV_{INST}.KE$ des empfangenden Kreditinstituts, welches dem entsprechenden Zertifikat des Empfängers entnommen wurde[8].

Allerdings wird die Chipkarte zur Berechnung von Zufallszahlen herangezogen, welche den Einmalschlüssel bilden.

Chipkarte			Endgerät	
			A1	Daten (FinTS-Nutzdaten und ggf. Signatur) für die Verschlüsselung bereitstellen
R2	RND	←	C2	Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge*
		→	A2	RND als Einmalschlüssel-Fragment KS_{LL} speichern
R3	RND	←	C3	Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge*
		→	A3	RND als Einmalschlüssel-Fragment KS_{LR} speichern
R4	RND	←	C4	Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge*
		→	A4	RND als Einmalschlüssel-Fragment KS_{RL} speichern
R5	RND	←	C5	Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge*
		→	A5	RND als Einmalschlüssel-Fragment KS_{RR} speichern
			A6	KS_{LL}, KS_{LR}, KS_{RL} und KS_{RR} zu KS konkatenieren und speichern
			A7	Daten mit KS (symmetrisch) verschlüsseln
			A8	KS mit $P_K.RECV_{INST}.KE$ (asymmetrisch) verschlüsseln
			A9	Verschlüsselungsdaten aufbauen und in FinTS-Nachricht einfügen
			A10	Verschlüsselte Daten als Binärdaten in Verschlüsselungsdaten einfügen
			A11	ggf. A1 bis A10 für weitere Nachrichten wiederholen
			A12	Verschlüsselte und signierte FinTS-Nachrichten zur weiteren Bearbeitung speichern

[8] [DIN-SIG4, Kapitel 6.10.1]: „If an enciphered document is sent, the card is not involved: the software computes the content encryption key, enciphers the document and finally enciphers the content encryption key by applying the receiver’s public key taken from the receiver’s KE certificate.“

♦ **Erläuterung**

1. Die Daten (FinTS-Nutzdaten und ggf. Signatur) für die Verschlüsselung werden bereitgestellt.
2. Mit dem Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge* lässt sich das Terminal eine Zufallszahl von der HBCI-Karte geben.

   Wenn das Kommando erfolgreich ausgeführt wurde, gibt die HBCI-Karte eine 8 Byte lange Zufallszahl als Antwortdatum aus, die als Einmalschlüssel-Fragment KS_{LL} gespeichert wird.
3. Mit dem Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge* lässt sich das Terminal eine zweite Zufallszahl von der HBCI-Karte geben, die als Einmalschlüssel-Fragment KS_{LR} gespeichert wird.
4. Mit dem Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge* lässt sich das Terminal eine dritte Zufallszahl von der HBCI-Karte geben, die als Einmalschlüssel-Fragment KS_{RL} gespeichert wird.
5. Mit dem Aufruf der ZKA-SIG-API-Funktion *zka_sig_get_challenge* lässt sich das Terminal eine vierte Zufallszahl von der HBCI-Karte geben, die als Einmalschlüssel-Fragment KS_{LL} gespeichert wird.
6. $KS_{LL,}$ $KS_{LR,}$ $KS_{RL,}$ und $KS_{RR,}$ werden zu KS konkateniert und gespeichert.
7. Die zu übertragenden Daten werden mit KS symmetrisch verschlüsselt (AES CBC-Mode, IV=0, ZKA-Padding).
8. Der Einmalschlüssel KS wird linksbündig mit Nullbits auf die Schlüssellänge aufgefüllt und anschließend mit dem öffentlichen Key-Encryption-Schlüssel $P_K.RECV_{INST}.KE$ des empfangenden Instituts, welches dem entsprechenden Zertifikat des Empfängers entnommen wurde, verschlüsselt. Das Ergebnis wird mit führenden Nullbits auf die Schlüssellänge erweitert.
9. Die Verschlüsselungsdaten werden aufgebaut und in die FinTS-Nachricht eingefügt.
10. Die verschlüsselten Daten als Binärdaten in die Verschlüsselungsdaten eingefügt.
11. Ggf. werden die Schritte 1 bis 10 für weitere Nachrichten wiederholt.
12. Die verschlüsselten und signierten FinTS-Nachrichten werden zur weiteren Bearbeitung gespeichert.

IV.1.3.2.5 FinTS-Dialog führen

Chipkarte			Endgerät			Kreditinstitut	
			A1	Benutzerkennung aus der bereits gelesenen Bankverbindung extrahieren			
		→ ←	M2	Nachricht signieren (siehe *IV.1.3.2.3 Nachrichten signieren*)			
			A3	Kommunikationszugang aus Bankverbindung herstellen			
			C4	Nachricht (beginnend mit Initialisierungsnachricht) senden	→ ←	R4	Antwortnachricht
			A5	falls Antwortnachricht verschlüsselt: Daten (Binärdaten in Verschlüsselungsdaten) und verschlüsselten Einmalschlüssel enc(KS) aus den Verschlüsselungsdaten für die Entschlüsselung bereitstellen			
		→ ←	M6	Ausführung der ZKA-SIG-API-Funktion *zka_sig_decrypt* zur Einmalschlüssel-Entschlüsselung, Resultat ist der Einmalschlüssel KS			
			A7	Daten mit Einmalschlüssel KS entschlüsseln.			
			A8	falls Kreditinstitutsnachricht signiert: Daten (Signatur, Nutzdaten) für Signatur-Prüfung bereitstellen			
		→ ←	M9	Signatur-Prüfung (siehe *IV.1.3.3.2 Signatur-Prüfung*)			
			A10	C4 bis M9 für alle weiteren FinTS-Nachrichten wiederholen			

IV.1.3.2.6 Signatur beenden

Chipkarte	

Endgerät	
M1	Aufruf der ZKA-SIG-API-Funktion *zka_sig_fini_signature_application*
M2	Aufruf der ZKA-SIG-API-Funktion *zka_sig_close*

- **Erläuterung**

1. Die ZKA-SIG-API-Funktion *zka_sig_fini_signature_application* wird ausgeführt. Diese Funktion setzt die ZKA-SIG-API in den Zustand „passiv“ und löscht die darin gespeicherten Werte.
2. Die ZKA-SIG-API-Funktion *zka_sig_close* gibt die Verbindung zum Kundenterminal wieder frei.

IV.1.3.3 Makros

IV.1.3.3.1 Signatur-Berechnung

Signaturen mit der Chipkarte werden im Rahmen der beiden Sicherheitsdienste „Authentication“ und „Non-Repudiation“ erzeugt.

- Sicherheitsdienst Authentication: Signatur mit Schlüssel SK.CH.AUTC/S (Client-Server-Authentifikations-Schlüssel)
- Sicherheitsdienst Non-Repudiation: Signatur mit Schlüssel S_K.CH.DS (Digitaler Signatur-Schlüssel)

Die tatsächliche Durchführung der Signatur durch die Chipkarte ist insbesondere an die Erfüllung von Zugriffsbedingungen geknüpft, hier sind dies insbesondere eine vorhergehende Benutzer-Authentifikation in Form der Verifikation

- des CSA-Passworts für die Erlaubnis zur Signatur mit dem Schlüssel SK.CH.AUTC/S
- der Signatur-PIN für die Erlaubnis zur Signatur mit dem Schlüssel SK.CH.DS

Durch einen in der Chipkarte personalisierten Parameter der Signatur-Anwendung [ZKASIG] wird dabei festgelegt, nach wie vielen elektronischen Signaturen spätestens die Benutzer-Authentifikation zu wiederholen ist. Eine Benutzer-Authentifikation wird bei Bedarf innerhalb der ZKA-SIG-API-Funktionen *zka_sig_digital_signature* bzw. *zka_sig_cs_authentication* durchgeführt.

Chipkarte			Endgerät	
R1	evtl. Hash-Wert	← →	M1	Hash-Wert HASH berechnen, optional unter Verwendung der ZKA-SIG-API-Funktion *zka_sig_hash*
R2a	Signatur	← →	M2a	Sicherheitsdienst Non-Repudiation: Aufruf der ZKA-SIG-API-Funktion *zka_sig_digital_signature* **oder:**
R2b	Signatur	← →	M2b	Sicherheitsdienst Authentication: Aufruf der ZKA-SIG-API-Funktion *zka_sig_cs_authentication*

♦ **Erläuterung**

1. Die Berechnung des Hash-Wertes erfolgt in der Regel außerhalb der Chipkarte (Hash-Algorithmus gemäß Vorgabe für den Sicherheitsdienst bzw. vom Kreditinstitut übermittelter BPD). Optional ist es auch möglich, den letzen Schritt oder alle Schritte der Hash-Wert-Berechnung durch die Chipkarte durchführen zu lassen. Diese Berechnung ist dann Bestandteil des Ablaufs der ZKA-SIG-API-Funktion *zka_sig_hash*. Der zu verwendende Hash-Algorithmus wird dabei in Form der zugehörigen OID übergeben:

   - OID = 2.16.840.1.101.3.4.2.1 für SHA-256

2a. Bei Verwendung des Schlüssels S_K.CH.DS (Sicherheitsdienst Non-Repudiation) wird die Signatur durch Aufruf der ZKA-SIG-API-Funktion *zka_sig_digital_signature* erzeugt. Die Auswahl des Signaturalgorithmus und Paddingverfahrens erfolgt gemäß Vorgabe für den Sicherheitsdienst bzw. vom Kreditinstitut übermittelter BPD. Die Signaturanwendung der Bankensignaturkarte bietet ab SECCOS 6 gemäß den für RAH spezifizierten Algorithmen RSASSA-PSS, Standard-RSA und SHA-256 das Verfahren „id-TA-RSA-PSS-SHA-256“ mit der OID = 0.4.0.127.0.7.2.2.2.1.4.

   Falls der Hash-Wert im vorangegangenen Schritt 1 durch die Chipkarte berechnet wurde, ist er noch in der Chipkarte gespeichert und braucht nicht erneut als Parameter des *zka_sig_digital_signature* übergeben zu werden.

2b. Bei Verwendung des Schlüssels S_K.CH.$AUT_{C/S}$ (Sicherheitsdienst Authentication) wird die Signatur durch Aufruf der ZKA-SIG-API-Funktion *zka_sig_cs_authentication* erzeugt. Die Chipkarte verwendet dabei intern ein Padding-Format gemäß PKCS#1 ([SECCOS-6, Kapitel 8.3.2.1][9]), wobei die Digest-Info nicht von der Chipkarte selbst erzeugt wird, sondern als aufbereiteter „Authentication-Input“ (= zu signierendes Datenfeld) übergeben werden muss.

[9] Auszug aus [SECCOS-6, Kapitel 8.3.2.1]: Falls der Authentication Input nicht zu lang ist, wird er zu einer Folge von N-1 Byte wie folgt formatiert:

Bezeichnung	Byte-Länge	Wert
Blocktyp	1	'01'
Paddingfeld (PS)	N-3-L	'FF...FF'
Separator	1	'00'
Datenfeld	L	Authentication Input (AI)

Der Authentication-Input ist wie folgt aufgebaut ([SECCOS-6, Kapitel 8.1.8.3.1]):

Tag	Länge	Wert	Erläuterung
'30'	'31'		Tag und Länge von SEQUENCE (SHA-256)
'30'	'0D'		Tag und Länge von SEQUENCE (SHA-256)
'06'	'09'	'60 86 48 01 65 03 04 02 01'	OID des SHA-256 (2 16 840 1 101 3 4 2 1)
'05'	'00'	-	TLV-Kodierung von NULL
'04'	'20'	'XX..XX'	Hash-Wert

Anmerkung: Die direkte Weiterverwendung eines eventuell im Chip berechneten und dort zwischengespeicherten Hash-Wertes ist bei der Signatur im Sicherheitsdienst „Authentication“ nicht möglich. Der Hash-Wert (als Ergebnis von Schritt 1) muss daher explizit als Aufrufparameter in der oben beschriebenen Form in Schritt 2 übergeben werden.

IV.1.3.3.2 Signatur-Prüfung

Die Bankensignaturkarte selbst unterstützt zurzeit keine Signatur-Prüfung[10]. Die Prüfung einer Signatur wird vom Kundenterminal-Makro “Überprüfen der Korrektheit der elektronischen Unterschrift” durchgeführt.

Die (mathematische) Korrektheit einer elektronischen Unterschrift wird überprüft, in dem sie mit dem entsprechenden öffentlichen Schlüssel entschlüsselt wird und das Ergebnis mit dem Hash-Wert über die signierten Daten verglichen wird. Der für die Überprüfung der elektronischen Signatur eingesetzte öffentliche Schlüssel liegt in dem Kundenterminal authentisch vor, falls die zu ihm gehörende Zertifikatshierarchie vorher ebenfalls in dem Kundenterminal überprüft wurde [KT-KONZEPT].

[10] [ZKASIG, Kapitel 1.1]: „Die DK-Chipkarte unterstützt [die] Signaturprüfung zur Zeit aus dem folgenden Grund nicht: Die Prüfung digitaler Signaturen, die mit beliebigen privaten Schlüsseln und/oder Algorithmen berechnet sind, würde voraussetzen, dass die Chipkarte X.509-Zertifikate auswertet. Dies ist gemäß Kapitel 16.1 von [DINSIG] zur Zeit nicht möglich.“