Elektronische Gesundheitskarte und Telematikinfrastruktur

Spezifikation
Identity Provider - Frontend

Version:	1.0.0
Revision:	241924
Stand:	30.06.2020
Status:	freigegeben
Klassifizierung:	öffentlich
Referenzierung:	gemSpec_IDP_Frontend

Dokumentinformationen

Änderungen zur Vorversion

Es handelt sich um die Erstversion des Dokumentes.

Dokumentenhistorie

Version	Stand	Kap./ Seite	Grund der Änderung, besondere Hinweise	Bearbeitung
1.0.0	30.06.20		initiale Erstellung des Dokuments	gematik

Inhaltsverzeichnis

1 Einordnung des Dokumentes

1.1 Zielsetzung

Die vorliegende Spezifikation definiert die Anforderungen zu Herstellung, Test und Betrieb des Produkttyps Identity Provider Frontend.

1.2 Zielgruppe

Das Dokument richtet sich an den Entwickler des Authenticators, welcher der Anbieter des IdP-Dienstes selbst oder ein direkt von ihm beauftragtes Subunternehmen ist. Außerdem richtet sich das Dokument an Entwickler von Anwendungsfrontends, womit Anwendungen oder Applikationen auf Endgeräten des Nutzers (Smartphone oder PC) gemeint sind.

1.3 Geltungsbereich

Dieses Dokument enthält normative Festlegungen zur Telematikinfrastruktur des deutschen Gesundheitswesens. Der Gültigkeitszeitraum der vorliegenden Version und deren Anwendung in Zulassungs- oder Abnahmeverfahren wird durch die gematik GmbH in gesonderten Dokumenten (z.B. Dokumentenlandkarte, Produkttypsteckbrief, Leistungsbeschreibung) festgelegt und bekannt gegeben.

Schutzrechts-/Patentrechtshinweis

Die nachfolgende Spezifikation ist von der gematik allein unter technischen Gesichtspunkten erstellt worden. Im Einzelfall kann nicht ausgeschlossen werden, dass die Implementierung der Spezifikation in technische Schutzrechte Dritter eingreift. Es ist allein Sache des Anbieters oder Herstellers, durch geeignete Maßnahmen dafür Sorge zu tragen, dass von ihm aufgrund der Spezifikation angebotene Produkte und/oder Leistungen nicht gegen Schutzrechte Dritter verstoßen und sich ggf. die erforderlichen Erlaubnisse/Lizenzen von den betroffenen Schutzrechtsinhabern einzuholen. Die gematik GmbH übernimmt insofern keinerlei Gewährleistungen.

1.4 Abgrenzungen

Spezifiziert werden in dem Dokument die von dem Produkttyp bereitgestellten (angebotenen) Schnittstellen. Benutzte Schnittstellen werden hingegen in der Spezifikation desjenigen Produkttypen beschrieben, der diese Schnittstelle bereitstellt. Auf die entsprechenden Dokumente wird referenziert (siehe auch Anhang 9).

Das vorliegende Dokument beschreibt ausschließlich die Schnittstellen, welche durch den Authenticator oder ein Anwendungsfrontend zu bedienen sind. Schnittstellen, welche durch den IdP-Dienst betrieben werden, sind im Dokument [gemSpec_IDP_Dienst], und

solche, welche durch Fachdienste zu bedienen sind, im Dokument [gemSpec_IDP_FD] beschrieben.

Die vollständige Anforderungslage für den Produkttyp ergibt sich aus weiteren Konzept- und Spezifikationsdokumenten, diese sind in dem Produkttypsteckbrief des Produkttyps IdP-Dienst [gemSpec_IDP_Dienst] verzeichnet.

Nicht Bestandteil des vorliegenden Dokumentes sind die Festlegungen zu den Themenbereichen, wie das Anwendungsfrontend mit den Fachdaten umgeht, welche proprietären Schnittstellen hierbei verwendet werden oder wie An- oder Abmeldeprozesse außerhalb der OIDC- bzw. OAuth 2.0 Funktionaliäten abgebildet werden.

1.5 Methodik

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID in eckigen Klammern sowie die dem RFC 2119 [RFC2119] entsprechenden, in Großbuchstaben geschriebenen deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet.

Sie werden im Dokument wie folgt dargestellt:
<AFO-ID> - <Titel der Afo>
Text / Beschreibung
[<=]

Dabei umfasst die Anforderung sämtliche zwischen Afo-ID und der Textmarke [<=] angeführten Inhalte.

Hinweis auf offene Punkte

Offene Punkten werden im Dokument in dieser Darstellung ausgewiesen.

2 Systemüberblick

Der Systemüberblick ist zentral im Dokument [gemSpec_IDP_Dienst] beschrieben und soll hier aus Redundanz- und Pflegegründen nicht wiederholt veröffentlicht werden. Im vorliegenden Dokument sind die Schnittstellen beschrieben, welche gemäß Abschnitt 3.2-Akteure auf die Rolle Resource Owner entfallen. Betrachtet werden das Authenticator-Modul und das generische Anwendungsfrontend am Beispiel E-Rezept.

Im aktuellen Stand des Dokumentes fehlen Festlegungen zur Integration von Primärsystemen in der Rolle der Authenticator Applikation.

Im aktuellen Stand des Dokumentes fehlen noch in Diskussion befindliche, Festlegungen zur Erweiterung des Integritätsschutzes des Discovery Documents sowie weiterer verwendeter Schlüssel. Die Standards sehen Verschlüsselungen vor, aber lassen die Methoden des Schlüsselaustauschs offen und die gematik ist noch in Abstimmung zu praktikablen Methoden.

Im aktuellen Stand des Dokumentes fehlen an einigen Punkten noch Verweise auf die zugrundeliegenden Operationen der Standards OpenID Connect und OAuth2.

3 Systemkontext

Das Frontend des Nutzers besteht aus zwei voneinander unabhängigen Komponenten, welche auch auf unterschiedlicher Hardware betrieben werden können. Hierbei handelt es sich um den Authenticator, welcher als vom IdP-Dienst bereitgestellte Anwendungskomponente die Freischaltung eines Zugriffs in Form der Willenserklärung einfordert, und das Anwendungsfrontend, auf welchem schlussendlich die vom Fachdienst bereitgestellten Informationen dargestellt werden. Will das Anwendungsfrontend auf einen Fachdienst zugreifen, muss dieser Zugriff durch den Authenticator genehmigt bzw. freigegeben werden. Bei dieser Freigabe wird der Nutzer aufgefordert, einen Nachweis darüber zu erbringen, dass er berechtigt ist, auf den Fachdienst zuzugreifen. Als Berechtigungsnachweis wird vom IdP-Dienst der Besitz der Smartcard (HBA, eGK oder SMC-B) erwartet und zudem, dass der Besitzer der Smartcard auch die Kenntnis der dazugehörigen PIN hat.

Will ein Nutzer mit seinem Anwendungsfrontend auf einen Fachdienst zugreifen, kann dieser Zugriff also nicht direkt erfolgen. Das Anwendungsfrontend trägt seinen Wunsch, auf den mit ihm verbundenen Fachdienst zugreifen zu wollen, über eine Umleitung (redirect) dem Authenticator vor und dieser organisiert die Klärung der Zugriffsberechtigung und Identifikation.

Da es sich bei den mit dem IdP-Dienst bereitgestellten Daten in der TI immer um im höchsten Maße schützenswerte Daten des Gesundheitswesens handelt, ist es obligatorisch, alle erdenklichen Maßnahmen umzusetzen, damit die darin enthaltenen Informationen weder Einbußen bei der Integrität erfahren noch ihre Vertraulichkeit beeinträchtigt wird. Es findet daher zusätzlich zur serverseitigen TLS-Verschlüsselung und -Signatur eine Ende-zu-Ende Verschlüsselung mit JSON-Schlüsselmaterial statt. Das bedeutet, dass der Absender einer Information die Daten asymmetrisch mit dem öffentlichen Schlüssel des endgültigen Empfängers verschlüsselt. In manchen Fällen werden die Daten vor der Verschlüsselung zudem mit dem privaten Schlüssel signiert. Die Verwendung von Zertifikaten oder gar Zertifikatsketten ist im Falle von JSON Web Token nicht vorgesehen.

Vorbedingung der folgenden Schritte ist, dass das Anwendungsfrontend sowie der zugehörige Authenticator [A_19904] bereits beim IdP-Dienst registriert sind.

Die Prozessschritte, welche später im Dokument erklärt werden, sind wie folgt:

1. Das Anwendungsfrontend erzeugt sich eigenes Schlüsselmaterial ("PUK_FRONT" &"PRK_FRONT").
2. Das Anwendungsfrontend meldet seine aktuelle URI und die ihres öffentlichen Schlüssels beim IdP-Dienst an.
3. Das Anwendungsfrontend erzeugt sich ein "SECRET" (Zufallswert) und bildet darüber den Hash.
4. Den Hash überträgt das Anwendungsfrontend über einen Redirect an den Authenticator.
5. Der Authenticator klärt mit dem IdP-Dienst, welche Informationen dieser benötigt, um ein Token auszustellen.
6. Der IdP-Dienst fordert das Claim und sogleich eine Challenge-Response über einen Zufall vom Authenticator.

7. Der Authenticator stellt das Claim zusammen und reicht die Challenge (Zufall) bei der Smartcard zur Signatur ein.
8. Der Nutzer gibt die im Claim angeforderten Daten frei und die PIN der Smartcard ein, um die Signatur auszulösen.
9. Der Authenticator sendet die Zustimmung und die signierte Response zusammen mit dem Hash an den IdP-Dienst.
10. Der IdP-Dienst erstellt einen "ACCESS_CODE" und sendet diesen an das Authenticator-Modul.
11. Der Authenticator sendet den "ACCESS_CODE" per Redirect an das Anwendungsfrontend.
12. Das Anwendungsfrontend reicht den "ACCESS_CODE" zusammen mit dem "SECRET" beim Token-Endpunkt ein.
13. Der Token-Endpunkt bildet über das "SECRET" selbst einen Hash und gibt bei Erfolg das "ID_TOKEN" aus.
14. Das Anwendungsfrontend reicht das gültige "ID_TOKEN" beim Fachdienst ein und erhält Zugriff.

Hinweis: Der Schritt 5 wird, wenn das Anwendungsfrontend ein Primärsystem ist, durch die Funktion "externalAuthenticate" am Konnektor bedient. Im Falle eines HBA oder einer eGK sind die Signaturfunktionen für eine nonQES-Signatur durch die Smartcard aufzurufen.

3.1 Akteure und Rollen

3.2 Akteure

Resource Owner

Resource Owner ist der Nutzer, welcher auf die beim Fachdienst (Protected Resource) für ihn bereitgestellten Daten zugreift.

Der Resource Owner verfügt über die folgenden Komponenten:

- Endgerät des Nutzers
- Authenticator-Modul
- Anwendungsfrontend

Resource Server [RFC7165 # section-5.2]

Der Resource Server ist der Dienst (Fachdienstanbieter), der dem Nutzer (Resource Owner) den Zugriff auf seine geschützten Ressourcen (Fachdienste) gewähren kann. Im Falle der Telematikinfrastruktur sind die darin angebotenen Dienste zwar in der Hoheit der Fachdienstanbieter, werden jedoch allen Besitzern eines gültigen "ID_TOKEN" angeboten.

Der Fachdienstanbieter, auf dem die geschützten Informationen (Protected Resources) liegen, ist in der Lage, auf Basis von "ID_TOKEN" darauf Zugriff zu gewähren. Ein solcher Token repräsentiert die delegierte Identifikation des Resource Owners.

Client

Der Client wird durch das Anwendungsfrontend (Relying Party) des Nutzers dargestellt, der auf geschützte Ressourcen (Fachdienste) des Resource Owners (Telematikinfrastruktur) zugreifen möchte. Das Anwendungsfrontend kann auf einem Server als Webanwendung, auf einem Desktop-PC oder mobilen Gerät (z.B. Smartphone) ausgeführt werden.

Authorization Server

Der Authorization Server authentifiziert den Resource Owner und stellt `"ID_TOKEN"` für den vom Resource Owner (Nutzer) erlaubten Anwendungsbereich (SCOPE) aus, welche dieser wiederum beim Fachdienst einreicht.

Der Authorization Server stellt hierfür die folgenden Schnittstellen bereit, welche durch eigene TLS-Zertifikate zu schützen sind. Alle der folgenden Endpunkte müssen über eigenes Schlüsselmaterial verfügen, damit diese auch physisch leichter voneinander getrennt werden können. Unter physischer Trennung ist der Betrieb auf unterschiedlicher Hardware zu verstehen.

- AUTH Authorization-Endpunkt
- TOKEN Token-Endpunkt
- INT Token Introspection-Endpunkt
- REV Token Revocation-Endpunkt
- INFO Userinfo-Endpunkt
- REGISTER Client Registration-Endpunkt
- DD Discovery Document-Endpunkt

Weitere Akteure im Kontext IdP-Dienst sind:

Protected Resource

Fachdienstschnittstelle

3.3 Nachbarsysteme

Als Nachbarsysteme des Anwendungsfrontend sind das Authenticator-Modul, der IdP-Dienst [gemSpec_IDP_Dienst] sowie die mit dem IdP-Dienst in Verbindung stehenden Fachdienste [gemSpec_IDP_FD] zu nennen.

4 Zerlegung des Produkttyps

Das Frontend lässt sich in zwei Module aufteilen, von welchen das Authenticator-Modul einmalig und das Anwendungsfrontend mehrfach vorkommen kann.

Das Authenticator-Modul bietet hierbei die quasi interne Schnittstelle zum IdP-Dienst an und wird für mobile Nutzer durch den Betreiber des IdP-Dienstes bereitgestellt.Die Schnittstelle wird hier als "Quasi-Intern" bezeichnet, weil der Anbieter des IdP-Dienstes das Authenticator-Modul oder den für die Funktionalität des Authenticator-Moduls verantwortlichen Quellcode beeinflussen kann bzw. selbst stellt. Es handelt sich somit um eine eigene vom IdP-Dienst nur ausgelagerte Funktionalität. Für Primärsysteme muss das Authenticator-Modul als Bestandteil des Primärsystems implementiert werden. Als Primärsysteme sollen hier PVS (Praxisverwaltungssystem), KVS (Krankenhausverwaltungssystem) und AVS (Apothekenverwaltungssystem) genannt sein. Die Beschreibung des Authenticator-Moduls findet in diesem Dokument statt, weil das Authenticator-Modul einen wesentlichen Bestandteil auf Seiten des Nutzers darstellt und somit nicht in der zentralen Providerzone der Telematikinfrastruktur betrieben ist. Authenticator-Modul und Anwendungsfrontend sind in diesem Zusammenhang als ortsveränderliche Komponenten auf unsicheren Endgeräten zu betrachten.

Das Anwendungsfrontend ist eine nicht spezifizierteSoftware, welche auf Fachdienste innerhalb der Telematikinfrastruktur (TI) zugreifen möchte, um dort auf die durch den Fachdienstangebotenen Fachdienstdaten lesend oder schreibend zuzugreifen.

5 Übergreifende Festlegungen

Rollenausschluss

Vorgaben bezüglich eines möglichen Rollenausschlusses werden im Rahmen des Vergabeverfahrens beschrieben.

Zugriff durch Mitarbeiter

Regelungen bezüglich des Zugriffs durch Mitarbeiter des IdP-Dienstes werden im Rahmen des Vergabeverfahrens beschrieben. Es ist davon auszugehen, dass Mitarbeiter des IdP-Dienstes keinen Zugriff auf schützenswerte Daten erhalten.

Dokumentationspflicht

Die Dokumentationspflicht betrifft neben dem IdP-Dienst auch das Frontend. Die Mitwirkungspflicht bezieht sich erstrangig auf das Authenticator-Modul und ist in [gemSpec_IDP_Dienst] im gleichnamigen Kapitel beschrieben.

Service Lokalisierung

Die URI des IdP-Dienstes wird durch die gematik an zentraler Stelle veröffentlicht und dem Entwickler des Anwendungsfrontends genannt, damit dieser diese fest in den Quellcode implementieren oder in Form einer Parameterdatei in seine Anwendung übernehmen kann.

A_19990 - Vorbedingung für das Authenticator-Modul
Das Authenticator-ModulMUSS sich das Discovery Document heruntergeladen, dieses erfolgreich ausgewertet, seine Anmeldung/Registrierung am Authorization Server erfolgreich abgeschlossen und eine gültige `"SUBJECT_SESSION"` mit dem Authorization Server etabliert haben, bevor ein Anwendungsfrontend diesen adressieren kann.
[<=]

Die Lokalisierung des IdP-Dienstes lässt sich aus dem Service Discovery Document des IdP-Dienstes und aus den Authorization Server-Meta-Informationen auslesen. Der Veröffentlichungspunkt des Discovery Document ist im zentralen Dokument [gemSpec_IDP_Dienst] im gleichnamigen Kapitel beschrieben.

5.1 Zertifikatsprüfung von Internet-Zertifikaten

Folgende Vorgaben gelten für die Prüfung von Internet-Zertifikaten.

A_20068 - Authenticator: Prüfung Internet-Zertifikate
Das Authenticator-Modul MUSS für die Prüfung des internetseitigen Zertifikats von Diensten der TI das Zertifikat auf ein CA-Zertifikat einer CA, die die "CA/Browser Forum Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates" (https://cabforum.org/baseline-requirements-documents/) erfüllt, kryptographisch (Signaturprüfung) zurückführen können. Ansonsten MUSS er das Zertifikat als "ungültig" bewerten.
Das Authenticator-Modul MUSS die zeitliche Gültigkeit des Zertifikats prüfen. Falls diese Prüfung negativ ausfällt, MUSS das Authenticator-Modul das Zertifikat als "ungültig" bewerten. **[<=]**

Hinweis: Der erste Teil von A_20068 ist gleichbedeutend damit, dass das CA-Zertifikat im Zertifikats-Truststore eines aktuellen Webbrowsers ist.

6 Funktionsmerkmale Authenticator-Modul

Das Authenticator-Modul ist externer Bestandteil der zentralen Dienstleistung des IdP-Dienstes und wird von diesem für mobile Endgeräte wie Smartphones und/oder PC/Laptops bereitgestellt. Die in Primärsystemen zum Einsatz kommende Form des Authenticator-Moduls wird ebenfalls durch den Anbieter des IdP-Dienstes bereitgestellt.

Die Bereitstellung erfolgt hierbei über die dem jeweiligen Betriebssystem üblicherweise zur Verfügung stehenden Portale in einer sicheren, für den Nutzer kostenfreien Form. Im Falle des Betriebssystems Android erfolgt die Bereitstellung im Google Play Store oder dem zukünftig für dieses Betriebssystem etablierten Portal.

Für das Betriebssystem Apple findet die ebenso sichere Bereitstellung im Apple App Store statt, wobei dem Nutzer auch hier keine Kosten durch den Abruf der Software entstehen dürfen.

Die funktionale Aufteilung findet daher in zwei Abschnitten statt, wenngleich beide Teile möglicherweise auf einem Gerät betrieben werden. Diese Aufteilung wird vorgenommen, da das Authenticator-Modul strikte Vorgaben bezüglich seines Verhaltens hat, welche durch das Anwendungsfrontend möglicherweise nicht umsetzbar wären. Die Betrachtung der einzelnen Softwarekomponenten erfolgt auch deswegen getrennt, da diese einen vollkommen anderen Funktionsumfang aufweisen.

Aufgabe des Authenticator-Moduls ist, die von einem Anwendungsfrontend zum Zugriff auf Fachdienste benötigten "ID_TOKEN" mit Zustimmung des Nutzers (Resource Owner) und nach eingehender Überprüfung dessen Identität am Authorization-Endpunkt zu beantragen und dem Anwendungsfrontend den "ACCESS_CODE" zukommen zu lassen, welchen das Anwendungsfrontend am Token-Endpunkt gegen das "ID_TOKEN" eintauschen kann.

Die für die Beantragung des "ID_TOKEN" notwendigen Informationen bekommt das Authenticator-Modul einerseits vom Anwendungsfrontend übergeben. Weitere Informationen bezieht das Authenticator-Modul mittels NFC-Schnittstelle von einer Smartcard. Die notwendige elektronische Signatur im Challenge-Response-Verfahren ruft das Authenticator-Modul ebenfalls von der Smartcard ab und fordert hierbei den Nutzer zur PIN-Eingabe auf. Weitere nicht normative Informationen hierzu finden sich im Kapitel 9.6.

6.1 Funktionsmerkmal des Authenticator-Moduls

Das Authenticator-Modul ist externer Bestandteil der Gesamtlösung IdP-Dienst und wird auf dem Endgerät des Nutzers (Frontend) betrieben. Das Authenticator-Modul wird durch den IdP-Dienst oder einen Dritten bereitgestellt und gepflegt.

6.1.1 Schnittstelle <I_XYZ>

Schnittstellen des Authenticator-Moduls sind diejenigen, an welchen Anfragen durch Anwendungsfrontends empfängt und jene, welche das Authenticator-Modul selbst verwendet, um mit dem Authorization-Endpunkt in Kontakt zu treten.

6.1.1.1 Schnittstellendefinition

A_19903 - Das Authenticator-Modul erstellt eigenes Schlüsselmaterial
Das Authenticator-ModulMUSS sich eigenes Schlüsselmaterial erstellen und den öffentlichen Teil des Schlüssels über den Authorization-Endpunkt veröffentlichen. Das Authenticator-Modul MUSS das Schlüsselmaterial gemäß Vorgaben der [gemSpec_Krypt] erstellen und verwalten. **[<=]**

A_19904 - Das Authenticator-Modul registriert sich am IdP-Dienst
Das Authenticator-ModulMUSS bei jedem Neustart seine aktuelle URI zusammen mit seinem öffentlichen Schlüssel `"PUK_APP"` beim IdP-Dienst registrieren [RFC7591#section-3.1]. **[<=]**

A_19905 - Überwachung der URI zur Laufzeit
Das Authenticator Modul MUSS zur Laufzeit überwachen, ob sich seine IP-Adresse und somit die URI verändert, um derartige Änderungen dem IdP-Dienst mitzuteilen.
[<=]

A_19906 - Das Authenticator-Modul ist nur einmalig am Authorization Server angemeldet
Das Authenticator-ModulDARF sich NICHT erneut anmelden, wenn es keine Änderungen der URI gab.
[<=]

A_19907 - Das Authenticator-Modul nimmt ausschließlich verschlüsselte Daten an
Das Authenticator-ModulMUSS eingehende Daten mit seinem privaten Schlüssel `"PRK_APP"` entschlüsseln.
[<=]

Nachdem das Authenticator-Modul die eingehenden Daten mit seinem privaten Schlüssel entschlüsselt hat, muss es die Herkunft, Integrität und Gültigkeit der Daten prüfen.

Die entschlüsselten Daten sind immer elektronisch signiert.

A_19908 - Eingehende Daten sind signiert
Das Authenticator-Modul MUSS die Signatur gegen den öffentlichen Schlüssel des Absenders "PUK_AUTH" prüfen. Liegt dem Authenticator-Modul der öffentliche Schlüssel des Absenders noch nicht vor,MUSS er diesen vom Authorization Server gemäß den Angaben im Discovery Document abrufen.
[<=]

Das Authenticator-Modul wird nur von denjenigen Anwendungsfrontends zur Authentifizierung herangezogen, welche ihm gegenüber vorher registriert wurden. Hierbei muss das vorher registrierte Authenticator-Modul beim IdP-Dienst als das für das Anwendungsfrontend zuständige eingetragen werden. Das Anwendungsfrontend erhält seinerseits bei der dynamischen Registrierung am IdP-Dienst einen eindeutigen Identifier, welcher im Authenticator-Modul einzutragen ist. Dadurch wird das Authenticator-Modul mit dem bestimmten Anwendungsfrontend auf dem bestimmten Endgerät verknüpft. Der IdP-Dienst hat somit das Wissen darüber, auf welchem Endgerät

sich das registrierte Authenticator-Modul des Nutzers befindet. Außerdem weiß der IdP-Dienst, welches Frontend mit dem Authenticator-Modul verbunden ist und wo sich dieses nach der dynamischen Anmeldung befindet. Will nun das Anwendungsfrontend einen Zugriff initiieren, sieht der IdP-Dienst nach, welches Authenticator-Modul zum vortragenden Anwendungsfrontend registriert ist, und leitet die Anfrage an dessen zuletzt dynamisch gemeldete URI um. Ist das Authenticator-Modul offline und somit nicht erreichbar, reagiert das Authenticator-Modul mit einem Hinweis und die Freischaltung des Fachdienstes erfolgt nicht. Ist das Authenticator-Modul online - also erreichbar -, leitet der IdP-Dienst den Redirect dorthin um und veranlasst das Authenticator-Modul, die weiteren Schritte in die Wege zu leiten. Da das Authenticator-Modul bereits vorher beim IdP-Dienst bekannt ist, kann auch dessen öffentlicher Schlüssel schon bereitgestellt werden.

Damit die anderen Akteure das dynamisch registrierte Authenticator-Modul im späteren Verlauf adressieren können, muss das Authenticator-Modul bzw. dessen URI `"URI_APP"` sowie die URI zum Downloadpunkt seines öffentlichen Schlüssels `"URI_PUK_APP"` bekanntgemacht werden.

A_20071 - Registrierung des Authenticator-Moduls
Das Authenticator-Modul MUSS sich beim Authorization Server an der URI des Client Registration-Endpunktes (siehe Discovery Document) gemäß [RFC8628] mit einem HTTP/1.1 POST Request registrieren.
[<=]

A_19911 - Zusammenstellen des Consents durch das Authenticator-Modul
Das Authenticator-Modul MUSS zusätzlich benötigte Attribute beim Einreichen des Token-Antrags ausschließlich von der mit dem Nutzer in Verbindung stehenden Smartcard (eGK, eHBA oder SMC-B) beziehen.
Das Authenticator-Modul MUSS das Attribut `"name"` der juristischen und natürlichen Personen sowie das Attribut `"sub"`beim Einreichen des Token-Antrags beim Authorization-Endpunkt entsprechend des Datenformates der Informationsquelle wie folgt befüllen:

Akteur	**Attribute `"name"`**	**Identifier `"sub"`**
Leistungserbringer (eHBA)	Vorname, Nachname	Telematik-ID
Leistungserbringerinstitution (SMC-B)	Organisationsbezeichnung	Telematik-ID
Versicherte (eGK)	Vorname, Nachname	KVNR

[<=]

A_19912 - Zusammenstellen des Consents durch das Authenticator-Modul (erweiterte Attribute)
Das Authenticator-Modul MUSS neben den immer einzureichenden Attributen ausschließlich diejenigen verwenden, welche es aus dem ihm vorliegenden nonQES-Signaturzertifikat (eGK, eHBA, SMC-B externalAuthenticate) heraus auslesen kann.**[<=]**

A_20069 - Das Authenticator-Modul nutzt ausschließlich Zertifikate als Attributquellen
Das Authenticator-Modul DARF NICHT andere Quellen als Zertifikate für Attribute verwenden.**[<=]**

A_19913 - Zertifikat und Signatur gehören zusammen
Das Authenticator-ModulMUSS sicherstellen, dass die im Consent freizugebenden Attribute mit denen aus dem zur Signatur verwendeten Signaturzertifikat der verwendeten Smartcard übereinstimmen.**[<=]**

A_20070 - Gültigkeitsprüfung von nonQES-Signatur nicht durch das Authenticator-Modul
Das Authenticator-Modul DARF die Gültigkeit des verwendeten Zertifikates bzw. der verwendeten Smartcard NICHT prüfen. **[<=]**

A_19914 - Zusammenstellen des Consents durch das Authenticator-Modul (Anreicherung der Attribute)
Deas Authenticator-ModulMUSS Attribute, welche es zusätzlich benötigt, über die NFC-Schnittstelle (Near Field Communication) oder einen kontaktbehafteten Smartcard-Reader (PC/SC) erlangen. **[<=]**

Attribute, welche diese Zertifikate enthalten, sind der [gemSpec_PKI] zu entnehmen.

A_20017 - Backchannel-Revocation am Endgerät des Nutzers
Das Authenticator-Modul MUSS eine Backchannel-Revocation durch aktives Beenden oder Deinstallation des Authenticator-Moduls ermöglichen. Der Nutzer MUSS das Authenticator-Modul hierzu aktiv beenden (Beenden erzwingen) [RFC8417 # section-2.1.2] oder das Authenticator-Modul deinstallieren.**[<=]**

Die Durchführung der Backchannel-Revocation ist im [RFC8417 # section-2.1.2] beschrieben.

A_20038 - Widerruf des ID_TOKEN durch das Anwendungsfrontend
Das Anwendungsfrontend MUSS, wenn es absichtlich gestoppt oder deaktiviert wird, das vorhandene `"ID_TOKEN"` und `"REFRESH_TOKEN"` aus dem RAM löschen.**[<=]**

Umsetzung

Für die Durchführung der Aufgaben des Authenticator-Moduls muss dieses auf Schnittstellen des IdP-Dienstes zurückgreifen. Diese Schnittstellen sind im Dokument [gemSpec_IDP_Dienst] beschrieben.

Außerdem übergibt das Authenticator-Modul dem Anwendungsfrontend den `"ACCESS_CODE"`, womit am Authorization-Endpunkt des IdP-Dienstes das `"ID_TOKEN"` abgerufen werden kann. Das Anwendungsfrontend kann – muss aber nicht – auf demselben Endgerät betrieben sein. In jedem Fall muss das Authenticator-Modul alle durch andere angebotenen Schnittstellen über deren URI ansprechen. Eine Inter-App-Kommunikation zwischen Authenticator-Modul und Anwendungsfrontend ist aktuell nicht vorgesehen, kann aber gegebenenfalls gemäß [RFC8252 # section-5] erfolgen.

6.1.1.2 Nutzung

Die im Abschnitt 6.1.1.1 beschriebenen Schnittstellen des Authenticator-Moduls werden durch das Anwendungsfrontend genutzt. Die Nutzung durch das Anwendungsfrontend erfolgt hierbei nicht immer vom gleichen Gerät aus, weswegen alle Schnittstellen des Authenticator-Moduls über dessen URI bereitgestellt werden müssen. Um zu verhindern, dass Dritte diese Schnittstelle missbräuchlich verwenden, muss aller Datenverkehr, der über diese Schnittstelle angenommen wird, durch Verschlüsselung und Signatur gesichert sein.

Ansonsten verhalten sich alle Schnittstellen des Authenticator-Moduls gemäß den Vorgaben aus den verwendeten Standards, wobei hier ausdrücklich darauf hingewiesen wird, dass jeglicher Datenverkehr zusätzlich zur TLS-Sicherung zusätzlich mit dem privaten Schlüssel zu signieren und mit dem öffentlichen Schlüssel der Gegenstelle zu verschlüsseln ist.

Die verwendeten Standards sind:

- RFC3986 (URI)
- RFC7009 (JSON REVOCATION)
- RFC7165 (JOSE)
- RFC7231 (HTTP)
- RFC7515 (JWS JSON SIGNATURE)
- RFC7516 (JWE JSON ENCRYPTION)
- RFC7517 (JWK JSON KEY)
- RFC7518 (JWE JSON ALGORITHM)
- RFC7519 (JWT JSON WEB TOKEN)
- RFC7520 (JOSE Protection)
- RFC7521 (Assertion Authorization)
- RFC7522 (Assertion SAML 2.0)
- RFC7523 (JSON TOKEN Profile)
- RFC6749 (Oauth2)
- RFC7591 (OAuth2 Dynamic Client Registration)
- RFC6750 (Oauth2 Bearer)
- RFC7636 (Oauth Proof Key for Public Client)
- RFC7662 (OAuth TOKEN INTROSPECTION)

6.1.2 Hardwaremerkmale

Das Authenticator-Modul greift auf die NFC-Schnittstelle des Nutzer-Endgerätes oder einen angeschlossenen Smartcard-Reader zu, um das nonQES-Signatur-Zertifikat auszulesen und gegen Einforderung der PIN-Eingabe im Challenge-Response-Verfahren eine Signatur auszulösen.

Das Endgerät des Nutzers muss in der Lage sein, entweder über NFC oder einen Smartcard-Reader mit der eGK oder dem eHBA zu kommunizieren. Die hierfür notwendige Middleware ist als gegebene Voraussetzung anzusehen und ist Bestandteil

des Betriebssystems bzw. wird zusammen mit dem Authenticator-Modul installiert und ist insofern durch den Anbieter des IdP-Dienstes bereitzustellen.

7 Funktionsmerkmale Anwendungsfrontend

Das Anwendungsfrontend ist eine unbestimmte Software-Komponente, welche darauf zugeschnitten ist, Fachdaten der Fachdienste der Telematikinfrastruktur zu verarbeiten. Die Verarbeitung tritt hier in Form von Erstellung, Änderung, Anzeige, Weitergabe und Löschung auf. Um den agierenden Akteur vor dessen Zugriff auf die Fachdienste zu identifizieren, ist der Besitz einer Smartcard und der damit verbundenen PIN notwendig.

Es liegt jedoch nicht im Fokus der Fachdienste, Identitätskontrollen durchzuführen und zu überprüfen, ob eine aktuell vorgetragene Identität valide und gültig ist. Aus diesem Grund wurde die Instanz IdP-Dienst geschaffen, deren alleinige Aufgabe es ist, bereits ausgegebene Identifikationsmittel auf deren aktuelle Gültigkeit und integere Form hin zu überprüfen und gleichzeitig sicherzustellen, dass der vortragende Akteur auch berechtigt ist, das Identifikationsmittel nutzen zu dürfen. Aus diesem Grund wird bei einigen Funktionalitäten im Zusammenhang mit der Smartcard eine PIN-Abfrage angefordert. Diese soll sicherstellen, dass Besitz (Smartcard) und Wissen (PIN) zur selben Zeit quasi am selben Ort zusammenkommen und willentlich eine bestimmte Aktion auslösen.

Das Anwendungsfrontend muss die in diesem Dokument beschriebenen Schnittstellen bedienen, um auf in der TI als zentrale Plattformleistung angebotene Fachdienste zugreifen zu können.

7.1 Anwendungsfrontend Vorbereitende Maßnahmen

A_19915 - Sicheres Schlüsselmaterial Anwendungsfrontend
Das Anwendungsfrontend MUSS bei jedem Start zur Laufzeit eigenes Schlüsselmaterial bestehend aus öffentlichem "PUK_FRONT" und privatem "PRK_FRONT" Teil gemäß der Vorgaben aus [gemSpec_Krypt] erzeugen.**[**<=**]**

A_19916 - Speicherung von Schlüsselmaterial durch das Anwendungsfrontend
Das Anwendungsfrontend DARF den privaten Schlüssel "PRK_FRONT" NICHT im oder außerhalb des Endgerätes speichern. Das Anwendungsfrontend DARF Schlüsselmaterial von außen NICHT verwenden.**[**<=**]**

A_19917 - Unterstützung durch hardwarenahe Algorithmen
Das Anwendungsfrontend SOLL einen auf der Anwendungsumgebung angebotenen Prozessorchip oder hardwarenahe Routinen wie "Adiantum" bei der Erzeugung des Schlüsselmaterials nutzen.**[**<=**]**

A_19918 - Alter von Schlüsselmaterial
Das Anwendungsfrontend DARF Schlüsselmaterial NICHT länger als 24 Stunden verwenden. Einmal verwendetes Schlüsselmaterial DARF NICHT erneut verwendet werden.**[**<=**]**

A_19919 - Wiederverwendung von Schlüsselmaterial
Um sicherzustellen, dass das vorliegende Schlüsselmaterial nicht wiederholt verwendet wird, MUSS das Anwendungsfrontend einen über den öffentlichen Schlüssel "PUK_FRONT"

gebildeten HASH-Wert mit dem Verwendungsdatum in einer lokalen Liste speichern.
Ist der HASH-Wert des aktuell verwendeten Schlüssels zu einem anderen Datum in der Liste eingetragen, MUSS das Schlüsselmaterial sofort erneuert werden.
Die Liste MUSS mindestens die HASH-Werte der verwendeten Schlüssel der letzten 10 Tage enthalten.**[**<=**]**

A_19920 - Feststellung und Beobachtung der eigenen URI (Authenticator-Modul)
Das Authenticator-Modul MUSS nach dem Start seine eigene URI zur Laufzeit feststellen und überwachen, ob diese sich während der Laufzeit ändert. **[**<=**]**

A_19921 - Meldung geänderter URI (Authenticator-Modul)
Das Authenticator-Modul MUSS Änderungen an seiner eigenen URI umgehend an den Authorization Server melden, um weiterhin erreichbar zu bleiben.**[**<=**]**

Hinweis: Die Änderung der vom Provider bereitgestellten IP-Adresse kann sich durch Timeout oder Netzwechsel jederzeit ändern.

A_19922 - Bereitstellung der "URI_FRONT" und des öffentlichen Schlüssels "PUK_FRONT"
Das Anwendungsfrontend MUSS nach dem Erzeugen des Schlüsselmaterials den aktuell zu verwendenden öffentlichen Schlüssel "`PUK_FRONT`" zusammen mit seiner aktuell gültigen URI beim Authorization-Endpunkt zur Registrierung einreichen [openid-heart-oauth2-1_0.html#rfc.section.2.2.4].
Hierzu sendet das Anwendungsfrontend einen HTTP/1.1 Post mit den Inhalten an den Authorization-Endpunkt.**[**<=**]**

A_19923 - Bildung von SECRET und HASH
Das Anwendungsfrontend MUSS zur Laufzeit ein SECRET (Zufallswert) bilden. Das SECRET MUSS eine Entropie von mindestens 128 Bit enthalten. Das Anwendungsfrontend MUSS über das SECRET einen HASH-Wert bilden. **[**<=**]**

Das zur Bildung des HASH-Wertes verwendete Verfahren (HASH-Algorithmus) ist dem Dokument [gemSpec_Krypt] zu entnehmen.

7.2 Anmelden des Anwendungsfrontend

Die Registrierung des Anwendungsfrontends erfolgt gemäß [RFC7591 # section-3].
Hierbei muss das Anwendungsfrontend die erforderlichen Informationen an den Client Registration-Endpunkt des IdP-Dienstes senden.

A_20072 - Inhalt des Registrierungs-Request Anwendungsfrontend
Bei der Registrierung MUSS das Authenticator-Modul die vom Authorization Server erwarteten Informationen als HTTP/1.1 POST Request übermitteln:

Parameter	Ausdruck
client_id	Ist der Client bereits registriert, überträgt er seinen Client-Identifier [RFC6749 # section-2.2].
scope	Liste der gewünschten Berechtigungen beim Fachdienst
description	Softwarebezeichnung des Anwendungsfrontends
version	Versionsnummer des Anwendungsfrontends (Produktversion gemäß [gemSpec_OM])
URI_PUK_FRONT	URI des öffentlichen Teil des selbst erzeugten Encryption Key (Verschlüsselungsschlüssel) des Anwendungsfrontends "URI_PUK_FRONT"

[<=]

Diese Registrierungsdaten sehen z.B. wie folgt aus:

```
POST /register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: server.example.com
{
  "redirect_uris": [
"https://client.example.org/callback",  "https://client.example.org/c
allback2"],
  "client_name": "My Example Client",
  "client_name#ja-Jpan-JP":
"\u30AF\u30E9\u30A4\u30A2\u30F3\u30C8\u540D",
  "token_endpoint_auth_method": "client_secret_basic",
  "logo_uri": "https://client.example.org/logo.png",
  "jwks_uri": "https://client.example.org/my_public_keys.jwks",
  "example_extension_parameter": "example_value"
}
```

Der Authorization Server registriert den Client, wenn die im Registrierungsprozess eingereichten Daten die benötigten Attribute enthalten und deren Wertebereiche den Vorgaben entsprechen.

Eine positive Registrierung [RFC7591 # section-3.2.1] quittiert der Client Registration-Endpunkt z.B. wie folgt, wobei ein "client_secret" als Attribut hier mit dem Wert "cf136dc3c1fc93f31185e5885805d" übergeben und zeitgleich dessen Gültigkeit hier mit "client_secret_expires_at": auf 2893276800 Sekunden nach "01.01.1970 T UTC 00:00:00Z" begrenzt wurde:

```
HTTP/1.1 201 Created
Content-Type: application/json
Cache-Control: no-store
```

```
Pragma: no-cache
{
"client_id": "s6BhdRkqt3",
"client_secret": "cf136dc3c1fc93f31185e5885805d",
"client_id_issued_at": 2893256800,
"client_secret_expires_at": 2893276800,
"redirect_uris": [
   "https://client.example.org/callback",
   "https://client.example.org/callback2"],
"grant_types": ["authorization_code", "refresh_token"],
"client_name": "My Example Client",
"client_name#ja-Jpan-JP":
    "\u30AF\u30E9\u30A4\u30A2\u30F3\u30C8\u540D",
"token_endpoint_auth_method": "client_secret_basic",
"logo_uri": "https://client.example.org/logo.png",
"jwks_uri": "https://client.example.org/my_public_keys.jwks",
"example_extension_parameter": "example_value"
}
```

Eine negative, also erfolglose, Registrierung wird je nach Fehlerursache (siehe [RFC7591 # section-3.2.2]) in etwa wie folgt quittiert:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "error": "invalid_redirect_uri",
  "error_description": "The redirection URI
  http://sketchy.example.com is not allowed by this server."
}
```

oder

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "error": "invalid_client_metadata",
  "error_description": "The grant type 'authorization_code' must be
  registered along with the response type 'code' but found only
  'implicit' instead."
}
```

A_19924 - Zuständiges Authenticator-Modul
Das Anwendungsfrontend MUSS seine Anfrage zu einem "ID_TOKEN" über einen Redirect beim IdP-Dienst an das ihm gegenüber registrierten Authenticator-Modul richten.
[<=]

Die Verknüpfung zwischen Authenticator-Modul und Anwendungsfrontend wird hierbei durch den IdP-Dienst gespeichert und aufgelöst.

A_19925 - Formulierung und Inhalte der Anfrage für ein ID_TOKEN
Das Anwendungsfrontend stellt den Antrag auf ein "ID_TOKEN" beim Authenticator-Modul in Form eines HTTP/1.1 POST Request und MUSS dabei mindestens die folgenden Attribute anführen:

- Programm-Bezeichnung
- Programm-Versionsnummer
- URI_FRONT
- URI_PUK_FRONT
- HASH-Wert des SECRET
- HASH-Algorithmus
- Fachdienstbezeichnung (Scope)

[<=]

A_19926 - Signatur der Anfrage auf ein ID_TOKEN
Das Anwendungsfrontend MUSS eine Anfrage für ein "ID_TOKEN" elektronisch signieren. Die Signatur MUSS hierbei mit dem privaten Teil des Schlüsselmaterials "PRK_FRONT" erfolgen. **[<=]**

A_19927 - Verschlüsselung der Anfrage auf ein ID_TOKEN
Das Anwendungsfrontend MUSS eine Anfrage auf ein "ID_TOKEN" nach der Signatur mit dem öffentlichen Schlüssel des Authorization-Endpunkt "PUK_AUTH" verschlüsseln, um die Informationen in der Anfrage auf das "ID_TOKEN" vor der Kenntnisnahme durch Dritte auf dem Transportweg zu schützen.**[<=]**

A_19928 - Reaktion auf Fehlercodes des Authenticator-Moduls
Das Anwendungsfrontend MUSS auf Fehlermeldungen des Authenticator-Moduls entsprechend reagieren. Eine exakte Form der Reaktion ist nicht vorgegeben [RFC6749 # section-1.7].**[<=]**

A_20085 - Fehlermeldungen des Anwendungsfrontend
Das Anwendungsfrontend MUSS leicht verständliche Fehlermeldungen ausgeben. Eine exakte Form der Fehlermeldung ist nicht vorgegeben [RFC6749 # section-1.7].**[<=]**

Wenn möglich, sollen dem Nutzer Hilfestellungen gegeben werden, anhand derer man die Wiederholung des Fehlers vermeiden kann.

A_19929 - Liste der zu verarbeitenden Fehler
Das Anwendungsfrontend MUSS für die folgenden Fehler Handlungen vorsehen:

- Dienst nicht bekannt (Der genannte Fachdienst ist dem Authenticator-Modul nicht bekannt.)
- Dienst nicht registriert (Der Dienst ist bekannt, jedoch noch nicht registriert.)
- Falsche Anfrage (Die Anfrage ist in der Zusammenstellung fehlerhaft.)

- PUK_FRONT veraltet (Der angebotene öffentliche Schlüssel ist älter als 48 Stunden.)
- PUK_AUTH verwendet (Der verwendete Verschlüsselungsschlüssel ist veraltet.)
- Device-ID fehlerhaft (bestehend aus UUID des Prozessors)

[<=]

A_19930 - Annahme des ACCESS_CODE
Das Anwendungsfrontend MUSS an der von ihm aktuell am Authorization-Endpunkt veröffentlichten URI den vom Authenticator-Modul übertragenen "ACCESS_CODE" annehmen.**[<=]**

A_19931 - Entschlüsselung des ACCESS_CODE
Das Anwendungsfrontend MUSS den eingehenden "ACCESS_CODE" mit seinem privaten Schlüssel "PRK_FRONT" entschlüsseln. **[<=]**

A_19932 - Signaturprüfung des "ACCESS_CODE"
Das Anwendungsfrontend MUSS die Signatur des "ACCESS_CODE" gegen den öffentlichen Schlüssel "PUK_AUTH" des Authorization-Endpunktes prüfen. **[<=]**

A_20086 - Abbruch bei Signaturfehler "PRK_AUTH" (Anwendungsfrontend)
Das Anwendungsfrontend MUSS den Vorgang abbrechen, wenn die Signatur des "ACCESS_CODE" veraltet, beschädigt oder mit einem nicht vorgesehenen Algorithmus erstellt ist. **[<=]**

A_20087 - Verhalten nach Abbruch wegen Signaturfehler "PRK_AUTH" (Anwendungsfrontend)
Das Anwendungsfrontend MUSS den Antrag auf ein "ID_TOKEN" erneut stellen, wenn es bei der Signaturprüfung am "ACCESS_CODE" zu Fehlern kommt.**[<=]**

A_19933 - Verständliche Fehlermeldungen
Das Anwendungsfrontend MUSS für den Nutzer verständliche Fehlermeldungen ausgeben.
Das Anwendungsfrontend MUSS dem Nutzer leicht verständliche Anweisungen geben, die zur Vermeidung der Wiederholung des Fehlers dienen, wenn der Fehler durch den Nutzer verursacht wurde.**[<=]**

A_19934 - Signatur des ACCESS_CODE
Das Anwendungsfrontend MUSS den empfangenen "ID_TOKEN" zusammen mit dem über das SECRET gebildeten HASH-Wert sowie der Information des bei der Bildung verwendeten Algorithmus mit der Kennzeichnung "ACCESS_CODE" zusammenstellen und signieren.
Für die Signatur MUSS das Anwendungsfrontend den privaten Teil des Schlüssels "PRK_FRONT" verwenden.**[<=]**

A_19935 - Verschlüsselung des ACCESS_CODE
Das Anwendungsfrontend MUSS die Übertragung des aus SECRET und "ACCESS_CODE" zusammengestellten Paketes mit dem öffentlichen Schlüssel "PUK_TOKEN" des Token-Endpunktes verschlüsseln.**[<=]**

A_19936 - Einmalige Übertragung des ID_TOKEN
Das Anwendungsfrontend MUSS den signierten und verschlüsselten "ACCESS_CODE" nach der Übertragung sicher löschen.
[<=]

A_20081 - Sicherung des ACCESS_CODE auf dem Transportweg
Das Anwendungsfrontend MUSS den "ACCESS_CODE" TLS-gesichert an den Token-Endpunkt als HTTP/1.1 GET Request übertragen.**[<=]**

A_19937 - Fehlermeldungen des Token-Endpunktes Anzeige
Das Anwendungsfrontend MUSS in der Lage sein, die vom Token-Endpunkt übertragenen Fehlermeldungen anzuzeigen.**[<=]**

A_20078 - Fehlermeldung des Token-Endpunktes Formatierung
Das Anwendungsfrontend MUSS die Formulierung und das Format der Fehlermeldungen sowie möglicher Hinweise zur Fehlervermeidung vom Token-Endpunkt übernehmen.**[<=]**

A_20079 - Ausfall der Fehlermeldung des Token-Endpunktes
Das Anwendungsfrontend MUSS im Falle eines Timeout selbständig eine Fehlermeldung generieren, wenn eine Fehlermeldung durch den Token-Endpunkt ausbleibt.**[<=]**

A_20080 - Signatur der Fehlermeldung des Token-Endpunktes
Das Anwendungsfrontend MUSS die Signatur der Fehlermeldungen des Token-Endpunktes mit dem privaten Schlüssel "PRK_FRONT" prüfen, nachdem diese entschlüsselt wurde.**[<=]**

A_19938 - Annahme des ID_TOKEN
Das Anwendungsfrontend MUSS das vom Token-Endpunkt ausgegebene "ID_TOKEN" als HTTP/1.1 Statusmeldung 200 verarbeiten. Das Anwendungsfrontend MUSS das "ID_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird.**[<=]**

A_19939 - Fachdienstkennung in den Meta-Informationen
Das Anwendungsfrontend MUSS aus den META-Informationen der HTTP/1.1-Nachricht die Informationen entnehmen, für welchen Fachdienst das "ID_TOKEN" zu verwenden ist.**[<=]**

A_19940 - Temporäre Speicherung der "ID_TOKEN"
Das Anwendungsfrontend MUSS das verschlüsselte "ID_TOKEN" aus dem RAM (Random Access Memory) sowie lokal gespeicherte Kopien desselben beim aktiven Beenden der Anwendung sicher löschen. **[<=]**

A_20077 - Temporäre Speicherung von "REFRESH_TOKEN"
Das Anwendungsfrontend MUSS vorhandene "REFRESH_TOKEN" aus dem RAM sowie lokal gespeicherte Kopien desselben beim aktiven Beenden der Anwendung sicher löschen.**[<=]**

A_19941 - Entschlüsselung des ID_TOKEN
Das Anwendungsfrontend DARF das eingehende "ID_TOKEN" NICHT entschlüsseln, da dieses zielgerichtet für den angesprochenen Fachdienst mit dessen öffentlichem Schlüssel "PUK_FD" verschlüsselt ist.**[<=]**

A_19943 - Weitergabe des ID_TOKEN
Das Anwendungsfrontend MUSS das "ID_TOKEN" an die vom Fachdienst im Discovery Document bekanntgegebene URI senden. Der Versand MUSS in Form eines HTTP/1.1 GET-Request ohne weitere Verschlüsselung erfolgen. **[<=]**

7.3 Abmelden des Anwendungsfrontends

Dem Anwendungsfrontend muss die Möglichkeit geboten werden, die gerade genutzte Session (z.B. durch Logout) aktiv zu beenden, um ein erneutes Anmelden des Nutzers zu erzwingen.

A_20093 - Abmelden durch das Anwendungsfrontend
Das Anwendungsfrontend MUSS die Möglichkeit bieten, sich gemäß [OIDC Backchannel v1.0 # LogoutToken] aktiv von der gerade verwendeten Session abzumelden, wodurch ein erneutes Anmelden mit erneuter PIN-Abfrage erforderlich wird.**[<=]**

8 Verteilungssicht

Das Anwendungsfrontend verteilt sich auf eine beliebige Anzahl von Endgeräten des Nutzers. Einzig das Authenticator-Modul soll ausschließlich auf einem Endgerät betrieben werden, da es sich ansonsten um unterschiedliche Profile handelt. Die Beschreibung der theoretisch möglichen Verwendung unterschiedlicher Authenticator-Profile bleibt hier aus, da durch das hier beschriebenen Authenticator-Modul ausschließlich ein einziger IdP-Dienst adressiert wird. Es ist somit für den IdP-Dienst der TI ausschließlich ein einziges Authenticator-Modul zugelassen. Die gematik behält sich das Recht vor hieran Änderungen vorzunehmen.

Die Verteilung ist somit:

1 Nutzer : 1 Authenticator-Modul : n Anwendungsfrontend

Teilsysteme sind möglicherweise auf einem Gerät oder mehreren Geräten vorhanden. Als Liste der möglichen Teilsysteme seien – nicht abschließend – die folgenden genannt:

- Mobiles Endgerät des Nutzers (z.B. Tablet-PC, Android Smartphone, Apple iPhone und weitere)
- Stationäres Endgerät des Nutzers (z.B. Konnektor, PVS, AVS, KVS oder PC)

In jedem Fall benötigt jeder Nutzer genau ein Endsystem, auf welchem der Authenticator installiert und eingerichtet ist.

A_20076 - Das Authenticator-Modul muss online sein
Das Authenticator-Modul MUSS online und für das Anwendungsfrontend erreichbar sein.**[<=]**

Hinweis: Anwendungsfrontends, die ein `"ID_TOKEN"` benötigen, müssen das Authenticator-Modul erreichen können, da sie nicht selbst beim IdP-Dienst ein Token beantragen können.

9 Anhang A – Verzeichnisse

9.1 Abkürzungen

Kürzel	Erläuterung
APDU	Application Protocol Data Unit (Nachricht, die auf der Applikationsebene zwischen einer Smartcard und der externen Welt ausgetauscht wird)
CAN	Card Access Number (Kartenzugriffsnummer, auf dem Kartenkörper aufgedruckte Ziffernfolge)
FCP	File Control Parameter (Liste mit Eigenschaften einer Datei, die auf einer Smartcard gespeichert ist)
NFC	Near Field Communication (Kommunikation im Nahfeld einer Antenne)
PACE	Password Authenticated Connection Establishment (Passwort authentisierter Verbindungsaufbau)
PICC	Proximity Integrated Circuit Card (kontaktlose Smartcard)
TI	Telematikinfrastruktur

9.2 Glossar

Begriff	Erläuterung
Funktionsmerkmal	Der Begriff beschreibt eine Funktion oder auch einzelne, eine logische Einheit bildende Teilfunktionen der TI im Rahmen der funktionalen Zerlegung des Systems.
Consent	Consent ist der Raum von Attributen, welche vom IdP-Dienst bezogen auf die im Claim des jeweiligen Fachdienstes eingeforderten Attribute zusammenfasst. Es besteht Einigkeit zwischen dem was gefordert wird und welche Attribute im Token bestätigt werden.

Das Glossar wird als eigenständiges Dokument (vgl. [gemGlossar]) zur Verfügung gestellt.

9.3 Abbildungsverzeichnis

9.4 Tabellenverzeichnis

9.5 Referenzierte Dokumente

9.5.1 Dokumente der gematik

Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument referenzierten Dokumente der gematik zur Telematikinfrastruktur. Der mit der vorliegenden Version korrelierende Entwicklungsstand dieser Konzepte und Spezifikationen wird pro Release in einer Dokumentenlandkarte definiert; Version und Stand der referenzierten Dokumente sind daher in der nachfolgenden Tabelle nicht aufgeführt. Deren zu diesem Dokument jeweils gültige Versionsnummern sind in der aktuellen, von der gematik veröffentlichten Dokumentenlandkarte enthalten, in der die vorliegende Version aufgeführt wird.

[Quelle]	**Herausgeber: Titel**
[gemGlossar]	gematik: Einführung der Gesundheitskarte – Glossar
[gemSpec_COS]	gematik: Spezifikation des Card Operating System (COS), Elektrische Schnittstelle
[gemSpec_HBA_ObjSys]	gematik: Spezifikation des elektronischen Heilberufsausweises, HBA Objektsystem
[gemSpec_HBA_ObjSys_G2.1]	gematik: Spezifikation des elektronischen Heilberufsausweises, HBA Objektsystem
[gemSpec_IDP_Dienst]	

[gemSpec_Krypt]	
[gemSpec_OM]	
[gemSpec_eGK_ObjSys_G2.1]	gematik: Spezifikation der elektronischen Gesundheitskarte, eGK-Objektsystem

9.5.2 Weitere Dokumente

Die weiteren zu beachtenden Dokumente sind im zentralen Dokument des Produkttyps IdP-Dienst [/wiki/Spezifikation/gemSpec_IDP_Dienst?selection=ML-104184] beschrieben.

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002 ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf
[TR-03110]	Technische Richtlinie 3110 des Bundesamtes für Sicherheit in der Informationstechnik https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03110/index_htm.html
[TR–03111]	Technical Guideline TR-03111, Elliptic Curve Cryptography, Version 2.10 – 2018-06-01 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03111/index_htm.html
[TR-03116-1]	Technische Richtlinie BSI TR-03116-1, Kryptographische Vorgaben für Projekte der Bundesregierung, Version: 3.20, Datum:21.09.2018, Status: Veröffentlichung, Fassung: September 2018 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03116/index_htm.html

9.6 Interaktionen mit Smartcards der TI

9.6.1 Einleitung

Zur Nutzung bestimmter Dienste der Telematikinfrastruktur ist eine Authentisierung der zugreifenden Person erforderlich. Dieses Dokument betrachtet ein Konzept für folgende User Story aus Kartensicht:

Ein Kartennutzer möchte eine eGK oder einen HBA über die kontaktlose Schnittstelle nutzen, um sich mittels Challenge- Response-Verfahren zu authentisieren.

9.6.2 Grober Ablauf aus Kartensicht

Das Authenticator-Modul verwendet folgenden Ablauf, wobei hier nur Interaktionen mit der Karte dargestellt werden:

1. Von der Karte wird das zur Identität gehörende X.509 Zertifikat gelesen.
2. Es wird eine Nutzerauthentisierung durchgeführt.
3. Die Identität signiert eine Challenge. Die Signatur ist die zugehörige Response.

Aus Sicht einer eGK oder eines HBA stellt sich die User Story wie folgt dar (siehe Abbildung 1):

1. Der User bringt die Karte in das Feld eines NFC-Kartenlesers. Dadurch bootet die Karte und es wird ein Kommunikationskanal etabliert.
2. Das Authenticator-Modul etabliert einen PACE-Kanal zur Karte.
3. Das Authenticator-Modul ermittelt den Kartentyp.
4. Das Authenticator-Modul wählt den privaten Schlüssel der zu verwendenden Identität aus.
5. Das Authenticator-Modul liest das X.509 Zertifikat aus.
6. Das Authenticator-Modul stößt eine Nutzerverifikation an.
7. Das Authenticator-Modul sendet ein Token zur Karte, welches von dieser signiert wird.
8. Dem Authenticator-Modul ist es möglich, beliebig viele weitere Token zur Karte zu schicken, um diese von der Karte signieren zu lassen.
9. Die Story endet damit, dass der User die Karte aus dem Feld des NFC-Kartenlesers entfernt, wodurch die Karte abgeschaltet wird.

Erläuterungen zu Abbildung 1: Betrachtet wird hier ein System, welches aus folgenden Komponenten besteht:

1. Auf einem mobilen Gerät (in Abbildung 1 nicht gezeigt) ist eine Software installiert, welche (unter anderem) das Authenticator-Modul enthält.
2. Das Authenticator-Modul greift mit Hilfe diverser Komponenten des mobilen Gerätes via NFC-Schnittstelle auf eine Karte mit kontaktloser Schnittstelle zu.
3. Die Karte mit kontaktloser Schnittstelle wird in Abbildung 1 mit PICC (Proximity Integrated Circuit Card) bezeichnet.
4. Das Authenticator-Modul wird (ganz grob) in folgende zwei Komponenten unterteilt:
   a. "Secure Messaging Layer", eine Komponente, welche einen PACE-Kanal zur PICC etabliert und dann für eine kryptographisch gesicherte Kommunikation verantwortlich ist.
   b. "other", der Rest des Authenticator-Moduls, der nicht zur Komponente Secure Messaging Layer gehört.

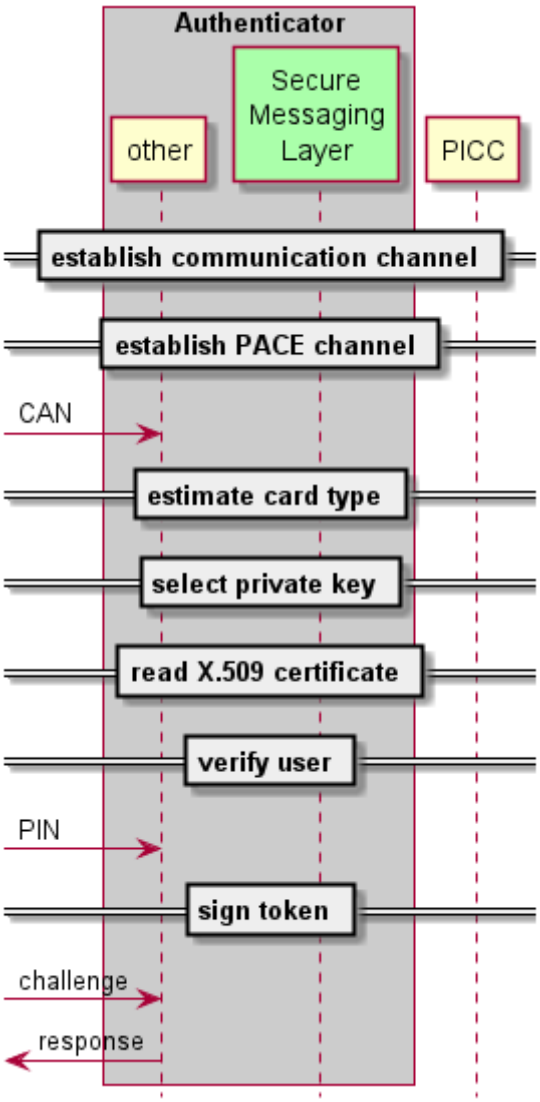

Abbildung 1: Überblick zum Ablauf

9.6.3 Ablauf im Detail

Die folgenden Unterkapitel schildern den Ablauf zum Signieren einer Challenge im Detail. Ziel der Darstellung in diesem Kapitel ist es, dass ein Entwickler in die Lage versetzt wird, auf Basis dieser Beschreibung (plus den Informationen aus verlinkten Dokumenten) die auf der kontaktlosen Nutzung von Smartcards der TI basierten Funktionen eines Authenticator-Moduls zu entwickeln.

9.6.3.1 Aufbau eines Kommunikationskanals zwischen Authenticator-Moduls und PICC

Wie ein Kommunikationskanal zwischen Authenticator-Modul und PICC etabliert wird, hängt von der Hard- und Software des Gerätes ab, auf welchem das Authenticator-Modul läuft und mit welchem die PICC verbunden wird.

9.6.3.2 Aufbau eines PACE-Kanals

Nach Aufbau eines Kommunikationskanals ist das Authenticator-Modul in der Lage, Kommandonachrichten an die PICC zu senden und korrespondierende Antwortnachrichten von dort zu empfangen. Da es sich bei NFC um eine Funkschnittstelle handelt, ist mit der Möglichkeit zu rechnen, dass Angreifer die Kommunikation belauschen oder beeinflussen. Aus Sicherheitsgründen sind Karten der TI so konfiguriert, dass sie (von wenigen Ausnahmen abgesehen) Funktionen über die kontaktlose Schnittstelle nur im Rahmen einer kryptographisch gesicherten Verbindung bereitstellen. Dem Stand der Technik entsprechend wird dabei PACE eingesetzt.

PACE (Password Authenticated Connection Establishment) bietet die Möglichkeit, selbst mit schwachen Passwörtern einen kryptographisch starken Kommunikationskanal zu etablieren. Das PACE-Protokoll wird kurz in [gemSpec_COS#15.4.2] beschrieben. Die dortige Beschreibung geht auf [TR-03110] zurück.

Der Aufbau eines PACE-Kanals gliedert sich grob in zwei Phasen:

1. Auswahl eines gemeinsamen Satzes von Parametern
2. Ablauf des PACE-Authentisierungsprotokolls

Anschließend sind beide Kommunikationspartner im Besitz starker kryptographischer Schlüssel, mit deren Hilfe sie die weitere Kommunikation absichern.

9.6.3.2.1 Auswahl eines gemeinsamen Satzes von Parametern

In [TR-03110] sind mehrere Varianten beschrieben, mittels eines (schwachen) Passwortes starke kryptographische Schlüssel zu vereinbaren. Drei dieser Varianten sind in [gemSpec_COS] verpflichtend enthalten. Welche Variante eine Karte der TI konkret unterstützt, ist in der Datei EF.CardAccess konform zu [TR-03110] codiert.

Eine allgemeine Funktion zum Aufbau eines PACE-Kanals würde die Daten aus EF.CardAccess auswerten. Derzeit wird in [gemSpec_eGK_ObjSys], [gemSpec_HBA_ObjSys] und [gemSpec_HBA_ObjSys_G2.1] nur genau eine PACE-Variante verwendet. Zudem ist zu erwarten, dass die derzeit verwendete Variante mindestens bis zum Jahre 2026 verwendet wird. Deshalb wird hier folgende Empfehlung für die Implementierung des Authenticator-Moduls ausgesprochen:

Das Authenticator-Modul verwendet die PACE-Variante "id-PACE-ECDH-GM-AES-CBC-CMAC- 128" und den Schlüssel SK.CAN mit der Schlüsselreferenz keyRef='02'.

Die verwendete PACE-Variante legt unter anderem die Domainparameter der zu verwendenden elliptischen Kurve fest.

9.6.3.2.2 Auswahl des passenden Schlüssels in der Karte

Vor dem Durchlauf des PACE-Protokolls ist auf der PICC der zugehörige Schlüssel SK.CAN auszuwählen. Dies geschieht mittels des Manage Security Environment Kommandos gemäß [gemSpec_COS#(N102.448)] mit den Parametern OID und keyRef gemäß der in 9.6.3.2.1 ausgewählten PACE-Variante. Gemäß der dort gegebenen Empfehlung gilt also:

- OID = " id-PACE-ECDH-GM-AES-CBC-CMAC- 128" und
- keyRef='02'.

Falls die PICC auf dieses Kommando NICHT mit dem Trailer '9000' = NoError antwortet, ist die PICC zu keiner der im Literaturverzeichnis aufgelisteten Objektsystemspezifikationen konform. In diesem Fall wird empfohlen, den Use Case abzubrechen.

9.6.3.2.3 Ablauf des PACE-Authentisierungsprotokolls

Der Ablauf des PACE-Authentisierungsprotokolls ist in [gemSpec_COS#CosA_8da] beschrieben. In der dortigen Abbildung sind drei Komponenten zu sehen, die wie folgt mit den Komponenten aus der unteren Abbildung "Sequenzdiagramm PACE-Authentisierung" korrespondieren:

Tabelle 1: Zusammenhang CosA_8da und Abbildung 2 in diesem Dokument

[COS#CosA_8da]	Abbildung 1	Anmerkung
COSb *PCD*	Secure Messaging Layer	Teilkomponente des NFC-Authenticator-Moduls, welche einen PACE-Kanal etabliert und anschließend für die geschützte Nachrichtenübertragung zur PICC sorgt.
Steuersoftware	Secure Messaging Layer	
COSa *PICC*	PICC	Karte der TI, eGK oder HBA

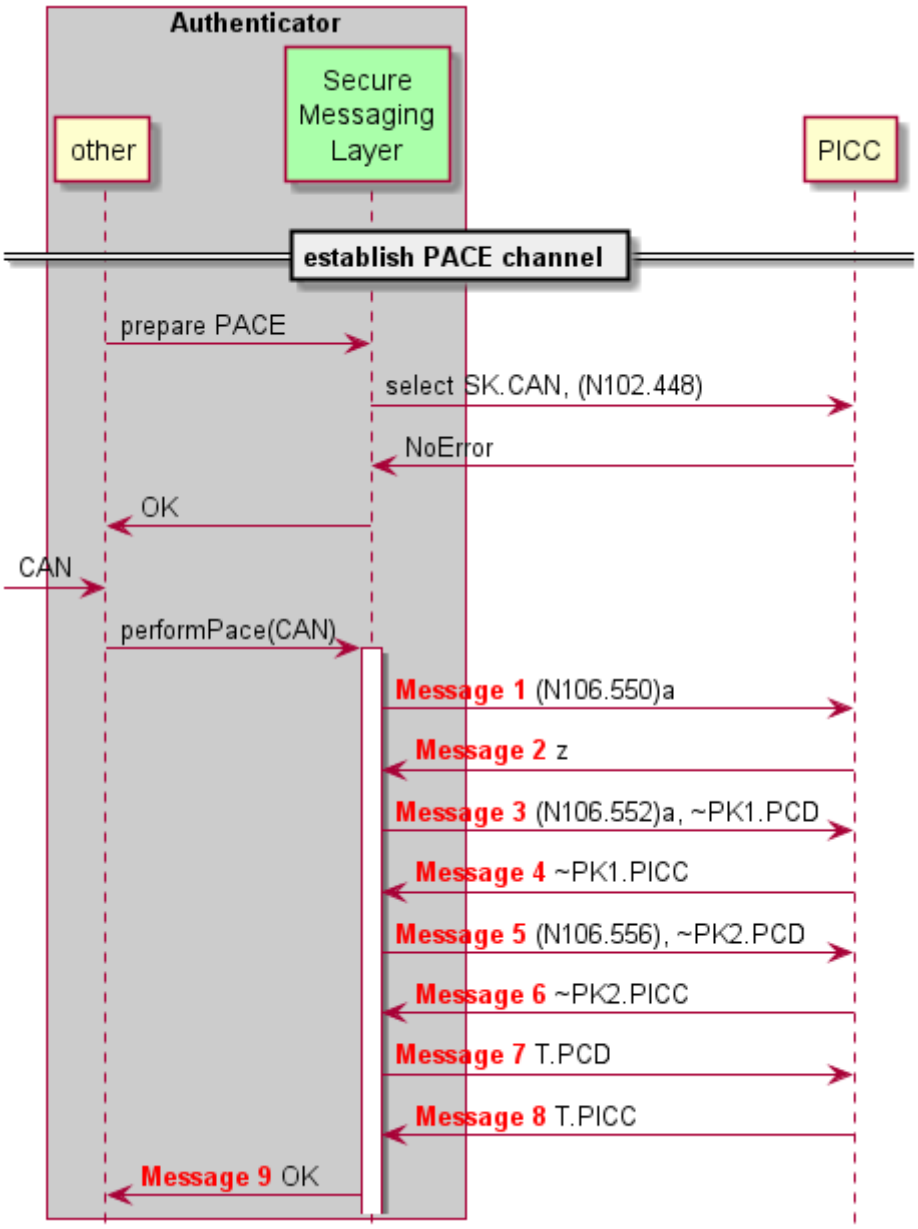

Abbildung 2: Sequenzdiagramm PACE-Authentisierung

Die PICC ist mit einer CAN (Card Access Number) ausgestattet. Die CAN ist auf dem Kartenkörper aufgedruckt. Die Komponente "Secure Messaging Layer" benötigt die CAN, um das PACE-Protokoll erfolgreich zu durchlaufen. Typischerweise wird die CAN einmalig vom Karteninhaber eingegeben. Es ist aber auch eine automatische Erkennung der CAN (etwa per Kamera) denkbar. Hier wird davon ausgegangen, dass der "Secure Messaging Layer" eine Komponente ist, die durch Übergabe der CAN zum Durchlaufen des PACE-Protokolls angestoßen wird.

Im Gutfall wird das PACE-Protokoll erfolgreich durchlaufen und im "Secure Messaging Layer" wurden dabei dieselben kryptographischen Schlüssel etabliert wie im PICC.

Im Schlechtfall scheitert einer der im Folgenden beschriebenen Schritte. Dann ist davon auszugehen, dass die aktuelle PICC nicht in der Lage ist, ein Token zu signieren. Gründe für das Scheitern umfassen unter anderem:

1. Die im "Secure Messaging Layer" verwendete CAN stimmt nicht mit der CAN überein, die in der PICC gespeichert ist.
2. Bei der PICC handelt es sich weder um eine eGK noch um einen HBA.

Im Folgenden werden die Nachrichten genauer beschrieben, die zwischen "Secure Messaging Layer" und PICC ausgetauscht werden.

Der "Secure Messaging Layer" erzeugt ein ephemeres Schlüsselpaar (~SK1.PCD, ~PK1.PCD), Details dazu in [gemSpec_COS#(N085.066)a.4].

Message 1: Der "Secure Messaging Layer" sendet ein General Authenticate Kommando gemäß [gemSpec_COS#(N106.550)a] zum PICC.

Message 2: Die Antwortdaten der PICC enthalten das Kryptogramm z.

Der "Secure Messaging Layer" errechnet mittels CAN und z die Zahl s gemäß [gemSpec_COS#(N085.066)b].

Message 3: Der "Secure Messaging Layer" sendet ein General Authenticate Kommando gemäß [gemSpec_COS#(N106.552)a] zum PICC, welches ~PK1.PCD enthält.

Message 4: Die Antwortdaten der PICC enthalten das Kryptogramm ~PK1.PICC.

Der "Secure Messaging Layer" errechnet mittels der Zahl s, dem Schlüssel ~SK1.PCD und ~PK1.PICC gemäß [gemSpec_COS#(N085.066)c] ephemere Domainparameter ~D und ein weiteres ephemeres Schlüsselpaar (~SK2.PCD, ~PK2.PCD).

Message 5: Der "Secure Messaging Layer" sendet ein General Authenticate Kommando gemäß [gemSpec_COS#(N106.556)] zum PICC, welches ~PK2.PCD enthält.

Message 6: Die Antwortdaten der PICC enthalten den Schlüssel ~PK2.PICC.

Der "Secure Messaging Layer" errechnet mittels ~D, SK2.PCD und ~PK2.PICC gemäß [gemSpec_COS#(N085.066)d] Sessionkeys k.MAC und k.ENC, sowie den MAC T.PCD.

Message 7: Der "Secure Messaging Layer" sendet ein General Authenticate Kommando gemäß [gemSpec_COS#(N106.560)] zum PICC, welches T.PCD enthält.

Message 8: Die Antwortdaten der PICC enthalten den MAC T.PICC

Der "Secure Messaging Layer" überprüft T.PICC gemäß [gemSpec_COS#(N085.066)e].

Falls kein Fehler auftrat, wird die Routine zum Etablieren des PACE-Kanals erfolgreich beendet. Damit liegen in der Komponente "Secure Messaging Layer" Sessionkeys vor. Die weitere Beschreibung geht davon aus, dass alle Kommandonachrichten des Authenticator-Moduls über die Komponente "Secure Messaging Layer" gesendet werden. Die Komponente "Secure Messaging Layer" schützt dabei Kommandonachrichten mit "Secure Messaging" gemäß [gemSpec_COS#13.2]. Die von der PICC empfangenen Antwortnachrichten sind kryptographisch gemäß [gemSpec_COS#13.3] geschützt und der "Secure Messaging Layer" prüft und entschlüsselt die Antwortnachrichten der PICC so, dass sie im Klartext und in der in [gemSpec_COS#14] beschriebenen Form zur Verfügung gestellt werden.

Es wird empfohlen, dass nach Aufbau des PACE-Kanals das Masterfile (MF) gemäß [gemSpec_COS#(N040.800)] selektiert wird. Wenn die PICC dieses Kommando erfolgreich durchgeführt hat, dann ist die PICC zuverlässig in einem Zustand, mit dem im Folgenden weitergearbeitet werden kann.

9.6.3.3 Ermitteln des Kartentyps

In der vorliegenden Fassung gilt die hier beschriebene Ermittlung des Kartentyps für [gemSpec_eGK_ObjSys], [gemSpec_HBA_ObjSys] und [gemSpec_HBA_ObjSys_G2.1]. Andere Kartentypen der TI (derzeit sind das SMC-B, gSMC-K und gSMC-KT) unterstützen das kontaktlose Interface nicht und sind deshalb irrelevant.

Unter anderem gibt es folgende Möglichkeiten, den Kartentypen für die hier relevanten Karten zu ermitteln:

1. Auslesen des ersten Rekords von EF.DIR gemäß [gemSpec_COS#(N066.100)].
   a. Vorteil: Die Antwortdaten sind kurz und weisen eindeutig auf den Kartentypen hin.
   b. Nachteil: Daten aus der Datei EF.DIR lassen sich über die kontaktlose Schnittstelle nur nach Aufbau eines PACE-Kanals auslesen.
2. Selektieren des Masterfiles (MF) ohne Applikation Identifier (AID) und Rückgabe der File Control Parameter (FCP) gemäß [gemSpec_COS#(N041.300)]. In diesem Fall ergäbe sich der Kartentyp aus dem Attribut AID, welches Teil der FCP ist.
   a. Vorteil: Funktioniert immer und über die kontaktlose Schnittstelle auch ohne PACE.
   b. Nachteile:
      i. Die FCP Daten sind je nach Implementierung sehr umfangreich und es ist möglich, dass zum vollständigen Empfang "extended length" erforderlich ist, weil die FCP Daten mehr als 256 Byte umfassen. Das Feature "extended length" war zumindest in der Vergangenheit für mobile Endgeräte problematisch.
      ii. Die FCP Daten werden als BER-TLV Struktur ausgegeben. Es erscheint nicht vorteilhaft, nur für die Interpretation der FCP Daten einen BER-TLV Parser zu implementieren.
3. Selektieren des Masterfiles (MF) mit Applikation Identifier (AID) gemäß [gemSpec_COS#(N040.800)], wobei dabei die *applicationIdentifier* Werte für eGK und HBA im Rahmen einer "Trial and Error"-Vorgehensweise durchzuprobieren wären. Die Karte antwortet entweder mit '6A82'=FileNotFound (Kartentyp und gewählter Wert von*applicationIdentifier* passen nicht zusammen) oder mit '900'=NoError (Kartentyp und gewählter Wert von*applicationIdentifier* passen zusammen).
   a. Vorteil: Funktioniert immer und über die kontaktlose Schnittstelle auch ohne PACE.
   b. Nachteil: "Trial and Error" verschlechtert die Performance, falls viele Versuche erforderlich sind.

Empfehlung: Erst nach Aufbau des PACE-Kanals ist eine kryptographisch gesicherte und damit zuverlässige Kommunikation möglich. Deshalb wird empfohlen, den Kartentyp durch Auslesen des ersten Rekords von EF.DIR zu ermitteln.

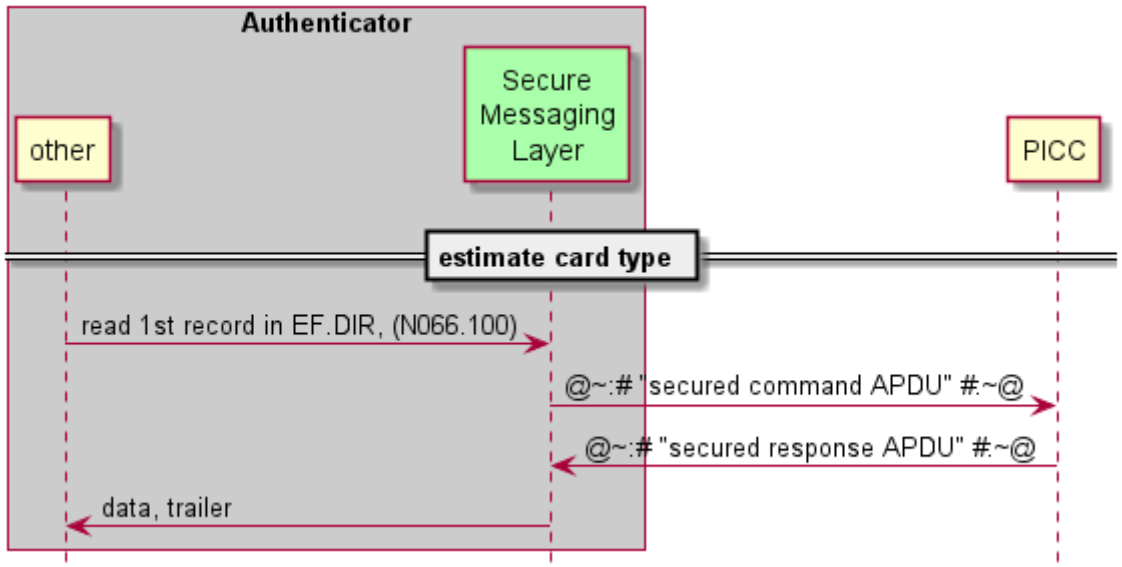

Abbildung 3: Ablauf zur Ermittlung des Kartentyps

Die Kommando APDU gemäß [gemSpec_COS#(N066.100) lautet in diesem Fall konkret: '00 B2 01F4 00'.

Die Antwort APDU enthält (im Erfolgsfall) Daten und einen Trailer: Fallunterscheidung: Falls

1. die Daten gleich '61094F07D2760001448000' sind, handelt es sich um eine eGK und der Trailer der Antwort APDU ist irrelevant.
2. die Daten gleich '61084F06D27600014601' sind, handelt es sich um einen HBA und der Trailer der Antwort APDU ist irrelevant.
3. der Trailer gleich '9000' = NoError ist und die Daten keinen der vorgenannten Werte enthalten, dann handelt es sich weder um eine eGK noch um einen HBA.
4. der Trailer gleich '6281' = CorruptDataWarning ist, dann wurden die Daten in der PICC möglicherweise verfälscht. Dieser Fall tritt äußerst selten auf. Falls entschieden wird, mit so einer PICC trotzdem weiter zu arbeiten, dann wird empfohlen, anhand der Länge und des Inhaltes der Daten zu entscheiden, mit welchem Wert der unterstützten Kartentypen die vorliegenden Daten die größere Ähnlichkeit aufweisen.

Nach Interpretation der Antwort APDU entscheidet das Authenticator-Modul, ob eine eGK, ein HBA oder ein unbekannter (nicht unterstützter) Kartentyp vorliegt.

9.6.3.4 Auswahl des privaten Schlüssels

Für den Kartentyp eGK sind private Schlüssel sowohl auf RSA-Basis als auch auf Basis elliptischer Kurven verfügbar. Dasselbe gilt für den Kartentyp HBA basierend auf [gemSpec_HBA_ObjSys_G2.1]. Lediglich für den Kartentyp HBA basierend auf [gemSpec_HBA_ObjSys] steht nur RSA-Kryptographie zur Verfügung.

Gemäß [TR–03116-1#3.2] ist die Moduluslänge von 2048 bit für RSA-Schlüssel nur bis Ende 2023 zulässig. Deshalb wird empfohlen, (sofern vorhanden) Schlüssel basierend auf elliptischen Kurven einzusetzen.

Nach dem bis hierher beschriebenen Ablauf aus 9.6.3.2 und 9.6.3.3 liegt zwar der Kartentyp fest, nicht aber die Information, ob es sich um einen HBA gemäß [gemSpec_HBA_ObjSys] oder [gemSpec_HBA_ObjSys_G2.1] handelt.. Unter anderem gibt es folgende Möglichkeiten herauszufinden, ob ein vorliegender HBA Schlüssel basierend auf elliptischen Kurven unterstützt:

1. Selektieren des ELC-Schlüssels, falls das gelingt, sind solche Schlüssel vorhanden.
   a. Vorteil: Einfach.
   b. Nachteil: "Trial and Error"-Methode. In9.6.3.3 wurde davon abgeraten, weil es einfache Ersatzmöglichkeiten gibt.
2. Auslesen von EF.ATR und Interpretieren des Inhaltes, hier Produktidentifikation des initialisierten Objektsystems.
   a. Vorteil: Eindeutige Aussage.
   b. Nachteil: Aufwendige Interpretation der BER-TLV-Strukturen.
3. Auslesen von EF.Version2 und interpretieren des Inhaltes, hier Produkttypversion des aktiven Objektsystems.
   a. Vorteil: Eindeutige Aussage.
   b. Nachteil: Aufwendige Interpretation der BER-TLV-Strukturen.

Zur Vereinfachung der Implementierung wird folgende Vorgehensweise empfohlen: Falls im Ablauf gemäß 9.6.3.3 als Kartentyp ermittelt wurde

1. eGK, dann wird *keyRef*='82' und*algId*='00' verwendet→PrK.CH.AUT.E256, signECDSA.
2. HBA, dann wird *keyRef*='86' und*algId*='00' verwendet→PrK.HP.AUT.E256, signECDSA.
3. HBA und die Selektion des privaten Schlüssels mit *keyRef*='86' schlägt fehl, dann wird*keyRef*='82' und*algId*='05' verwendet→PrK.CH.AUT.R2048, signPSS.

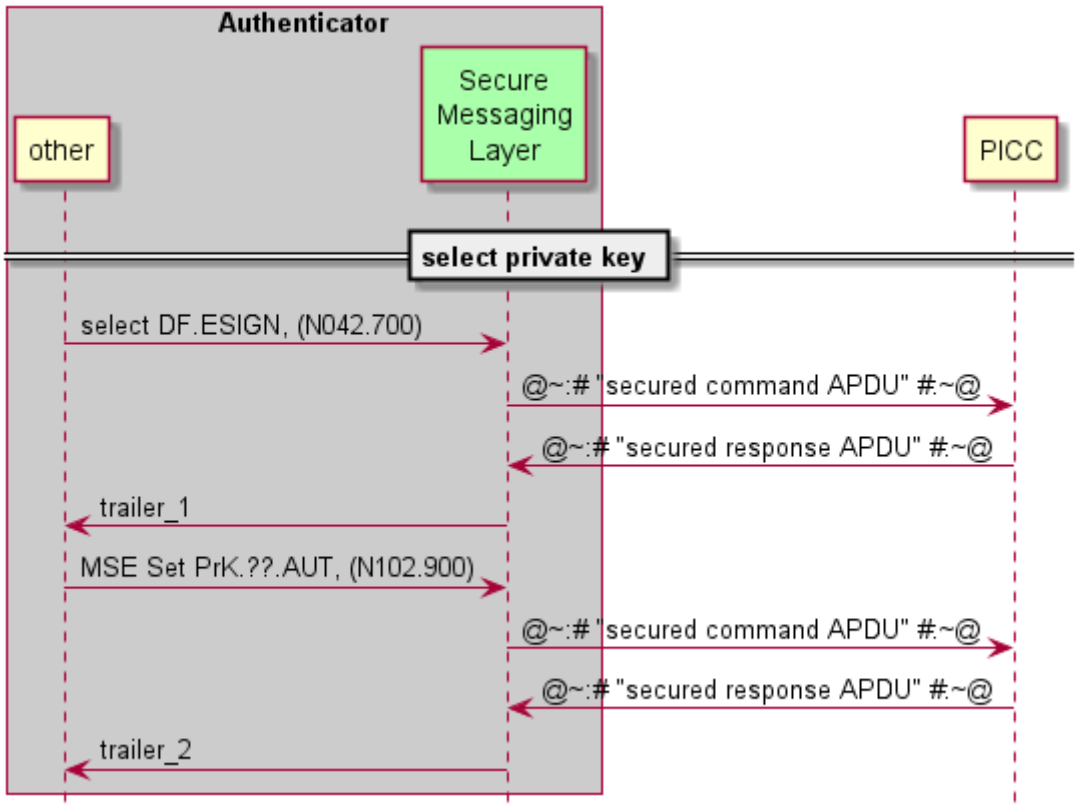

Abbildung 4: Ablauf zur Selektion des privaten Schlüssels

Vor der Selektion des privaten Schlüsselobjektes ist zunächst die passende Anwendung auf der PICC zu selektieren, in diesem Fall das DF.ESIGN. Die Kommando APDU gemäß [gemSpec_COS#(N042.700)] lautet in diesem Fall (sowohl für eGK als auch für HBA): '00 A4 040C 0A A000000167455349474E'. Die zugehörige Antwort APDU enthält lediglich "trailer_1". Falls "trailer_1" ungleich '9000' = NoError ist, dann handelt es sich weder um eine eGK noch um einen HBA.

Nach Selektion der Anwendung DF.ESIGN wird gemäß der Empfehlung der private Schlüssel gemäß [gemSpec_COS#(N102.900)] selektiert. Die Kommando APDU lautet dann:

Kartentyp gemäß	***keyRef***	***algId***	**Kommando APDU**
[gemSpec_eGK_ObjSys_G2.1]	'82'	'00'	'00 22 41B6 06 840182800100'
[gemSpec_HBA_ObjSys_G2.1]	'86'	'00'	'00 22 41B6 06 840186800100'
[gemSpec_HBA_ObjSys]	'82'	'05'	'00 22 41B6 06 840182800105'

Die zugehörige Antwort APDU enthält lediglich "trailer_2". Falls "trailer_2" gleich '9000' = NoError ist, dann wurde der private Schlüssel erfolgreich selektiert. Falls "trailer_2" ungleich '9000' ist, dann enthält die Anwendung DF.ESIGN keinen privaten Signaturschlüssel mit der angegebenen*keyRef* und *algId*.

9.6.3.5 Lesen des X.509 Zertifikates

Das zum privaten Schlüsselobjekt zugehörige X.509 Zertifikat enthält Informationen, die für die Validierung einer Signatur erforderlich sind. Deshalb ist die Kenntnis des X.509 Zertifikates signifikant. Die Informationsmenge im X.509 Zertifikat ist so groß (bis zu 1.900 Byte), dass sie nicht bei allen zulässigen Implementierungen mit einem einzigen Kommando aus der PICC auslesbar ist. Hinzu kommt, dass es für mobile Geräte eine Obergrenze in der Datenmenge gibt, die im Rahmen eines Kommando-Antwortpaares austauschbar sind. Für das Auslesen des X.509 Zertifikates sind also zwei Puffergrößen relevant:

1. Puffergröße der PICC: Der Wert der Puffergröße ließe sich aus der Datei EF.ATR auslesen.
2. Puffergröße des mobilen Endgerätes: Mir ist nicht bekannt, wie dieser Wert sicher zu ermitteln wäre.

Empfehlung: Es wird empfohlen, beim Auslesen des X.509 Zertifikates eine Blockgröße von 223 zu wählen. Damit ist die Größe einer gesicherten Antwortnachricht kleiner als 256 Byte. Der Wert von 256 Byte erscheint ein sicherer Wert für die Puffergröße mobiler Endgeräte. Dieser Wert ist erheblich kleiner als die 1033 Byte, die gemäß [gemSpec_COS#(N029.890)a.4] mindestens zu unterstützen sind.

Alles in allem wird das X.509 Zertifikat also in mehreren Blöcken zu je (empfohlenen) 223 Byte gelesen. Dabei ist zu unterscheiden zwischen dem ersten Lesekommando und allen weiteren Lesekommandos. Es wird empfohlen, im ersten Lesekommando gemäß [gemSepc_COS#(N051.500)], also Read Binary Kommando und mit *shortFileIdentifier* vorzugehen. Alle weiteren Lesekommandos verwenden [gemSpec_COS#(N051.100)], also Read Binary Kommando ohne *shortFileIdentifier*.

Im ersten Lesekommando wird *offset* = 0 gewählt. Bei allen weiteren Lesekommandos (im Sinne einer do-while-Schleife) wird der *offset* auf die Anzahl der bislang ausgelesenen Byte gesetzt. Die do-while-Schleife bricht ab, wenn die Antwortnachricht auf ein Lesekommando keine Antwortdaten mehr enthält, sondern nur noch den obligatorischen Trailer. Welchen Wert dieser Trailer hat, ist für den weiteren Verlauf irrelevant, sofern die bis dahin ausgelesenen Daten (konkateniert) ein valides X.509 Zertifikat ergeben. Die Validierung des X.509 Zertifikates ist nicht Gegenstand dieses Dokumentes.

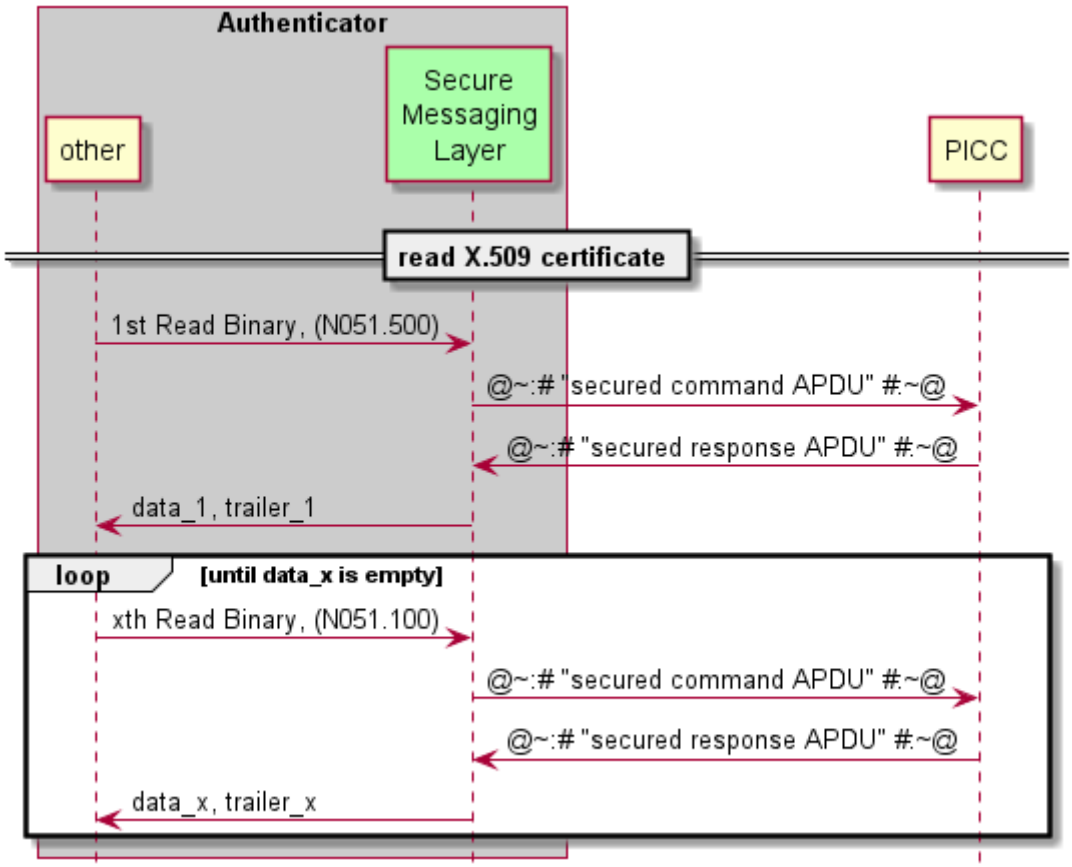

Abbildung 5: Ablauf zum Auslesen des X.509 Zertifikates

Das erste Lesekommando wählt gleichzeitig die zu lesende Datei aus. Deshalb enthält das erste Lesekommando einen *shortFileIdentifier*. Der *shortFileIdentifier* ist in Abhängigkeit vom Kartentypen gemäß 9.6.3.3 und des ausgewählten privaten Schlüssels gemäß 9.6.3.4 zu wählen.

Kartentyp gemäß	***privater Schlüssel***	***SFI***	**Kommando APDU**
[gemSpec_eGK_ObjSys_G2.1]	PrK.CH.AUT.E256	4	'00 B0 8400 DF'
[gemSpec_HBA_ObjSys_G2.1]	PrK.HP.AUT.E256	6	'00 B0 8600 DF'
[gemSpec_HBA_ObjSys]	PrK.HP.AUT.R2048	1	'00 B0 8100 DF'

Für die weiteren Lesekommandos ist ein *offset* zu wählen, der gleich der Anzahl N der bislang ausgelesenen Byte entspricht. N in hexadezimaler Darstellung besteht aus einem most-significant-byte MSByte_N und einem least-significant-byte LSByte_N. Die folgende Tabelle zeigt die (ungesicherte) Kommando APDU für alle weiteren Lesekommandos

(diese sind unabhängig vom Kartentypen und unabhängig vom ausgewählten privaten Schlüsselobjekt):

zweites Lesekommando	'00 B0 00DF DF'
drittes Lesekommando	'00 B0 01BE DF'
viertes Lesekommando	'00 B0 029D DF'
fünftes Lesekommando	'00 B0 03C7 DF'
sechstes Lesekommando	'00 B0 04B5 DF'
siebtes Lesekommando	'00 B0 053A DF'
achtes Lesekommando	'00 B0 0619 DF'
neuntes Lesekommando	'00 B0 06F8 DF'

Aus dem obigen Text und Abbildung 5 geht die Empfehlung hervor, die Schleife dann abzubrechen, wenn die Antwortnachricht keine Daten (oder, was dasselbe ist, leere Daten) enthält. Diese Vorgehensweise hat den Vorteil, dass der Wert des Trailers irrelevant ist. Typischerweise wird bei dieser Vorgehensweise ein Kommando zu viel geschickt. Falls die Anzahl an Bytes im X.509 Zertifikat kein Vielfaches der Blockgröße (hier 223) ist, dann gibt die PICC im Trailer anfangs '9000' = NoError zurück, dann bei vorhandenen Daten (also Anzahl Bytes in data_x größer null) '6282' = EndOfFileWarning, und wenn dann doch noch weiter gelesen wird '6B00' = OffsetTooBig zurück. Für den (eher untypischen) Fall, dass die Anzahl Bytes im X.509 Zertifikat ein Vielfaches der gewählten Blockgröße ist, wird nie '6282' = EndOfFileWarning gemeldet. Eine Implementierung, welche mit der minimalen Anzahl an Lesebefehlen auskommen will, hat dies zu berücksichtigen.

Weil das auszulesende X.509 Zertifikat als ASN.1 Codierung vorliegt, ist es möglich, die genaue Anzahl der Bytes durch Analyse der ersten ausgelesenen Bytes zu ermitteln. Es erscheint nicht sinnvoll, diesen Aufwand zu treiben.

9.6.3.6 Benutzerverifikation

Die Verwendung des privaten Schlüsselobjektes ist erst nach einer Benutzerverifikation möglich. Es wird empfohlen, die Benutzerverifikation erst nach Validierung des X.509 Zertifikates durchzuführen, weil eine Benutzerverifikation bei nicht validem X.509 Zertifikat überflüssig erscheint. Für die im Rahmen dieses Dokumentes betrachteten privaten Schlüsselobjekte ist die Benutzerverifikation nach einem Aktivieren der PICC nur

genau einmal notwendig. Anschließend lassen sich die hier behandelten privaten Schlüsselobjekte beliebig oft verwenden.

Für die Benutzerverifikation ist dem Authenticator-Modul das Geheimnis (also die PIN) bekanntzugeben. Typischerweise wird die PIN vom Benutzer eingegeben. Für die Benutzerverifikation ist die Zahlenfolge der PIN in einen Format-2-PIN-Block gemäß [gemSpec_COS#(N008.100)] umzuwandeln. Die folgende Tabelle enthält einige Beispiele für eine derartige Umwandlung.

PIN	Format-2-PIN-Block
123456	26123456FFFFFFFF
7531246	277531246FFFFFFF
87654321	2887654321FFFFFF

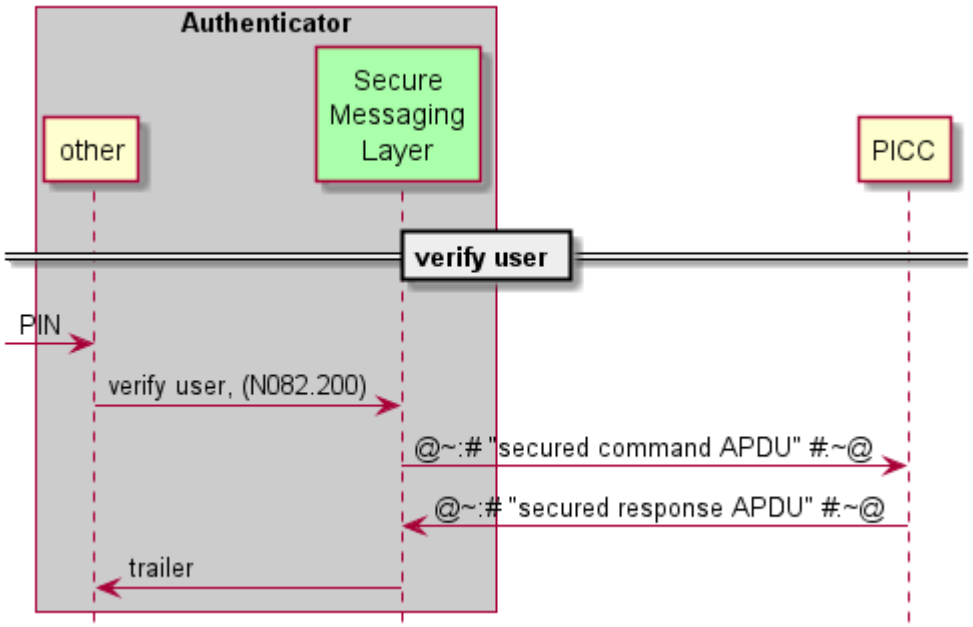

Abbildung 6*: Ablauf zur Verifikation des Benutzers*

Für die Benutzerverifikation ist nur eine Kommandonachricht erforderlich. Welches Passwortobjekt dabei verwendet wird, hängt vom Kartentypen ab.

Kartentyp gemäß	Password	*pwdId*	Kommando APDU
[gemSpec_eGK_ObjSys_G2.1	MRPIN.home	2	'00 20 0002 08 Format-2-PIN-Block'
[gemSpec_HBA_ObjSys_G2.1]	PIN.CH	1	

[gemSpec_HBA_ObjSys]			'00 20 0001 08 Format-2-PIN-Block'

Falls die Antwort APDU den Trailer '9000' = NoError enthält, lassen sich mit dem privaten Schlüsselobjekt Signaturen erstellen.

9.6.3.7 Signieren

Typischerweise sind per digitaler Signatur geschützte Artefakte mitunter sehr groß (etwa einige Megabyte oder auch Gigabyte). Wegen der begrenzten Bandbreite ist es nicht sinnvoll, die kompletten Artefakte zu einer Karte zu übertragen. Technisch bedeutet dies, dass der Signaturvorgang arbeitsteilig abläuft. Sicherheitstechnisch unbedenkliche Operationen laufen außerhalb der Karte ab und nur die sicherheitskritischen Operationen werden in der Karte ausgeführt. Dazu wird an der Schnittstelle zur Karte ein Zwischenergebnis der Signaturberechnung übergeben.

9.6.3.7.1 Signaturen mit dem Algorithmus signPSS = RSASSA-PSS

Der Algorithmus signPSS = RSASSA-PSS ist in [PKCS #1] Kapitel 8.1.1 beschrieben. Dabei wird die zu signierende Nachricht *M* zunächst gemäß EMSA-PSS-Encode codiert. Das Codierverfahren EMSA-PSS-Encode ist in [PKCS #1] Kapitel 9.1.1 beschrieben. Außerhalb der Karte werden dabei die Schritte gemäß [PKCS #1] Kapitel 9.1.1 Steps 1 und 2 mit dem Hash-Algorithmus SHA-256 durchgeführt. Als Ergebnis liegt der Hashwert *mHash* vor, der in 9.6.3.7.3 weiterverarbeitet wird.

9.6.3.7.2 Signaturen mit dem Algorithmus signECDSA

Der Algorithmus signECDSA ist in [TR–03111#4.2.1.1] beschrieben. Dort wird in Actions 5 der Hashwert $H_\tau(M)$ verwendet. Im vorliegenden Fall ist dieser Wert wie folgt zu berechnen: $H_\tau(M)$ = *mHash* = SHA-256 Hashwert der Nachricht *M*. Als Ergebnis liegt der Hashwert *mHash* vor, der in 9.6.3.7.3 weiterverarbeitet wird.

9.6.3.7.3 Signiervorgang

Eine Nachricht *challenge* wird signiert. Hier wird davon ausgegangen, dass dem Authenticator-Modul die zu signierende Nachricht *challenge* übergeben wird. Zunächst berechnet das Authenticator-Modul gemäß 9.6.3.7.1 oder 9.6.3.7.2 den SHA-256 Hashwert zu *challenge* gemäß *mHash* = SHA-256(*challenge*).

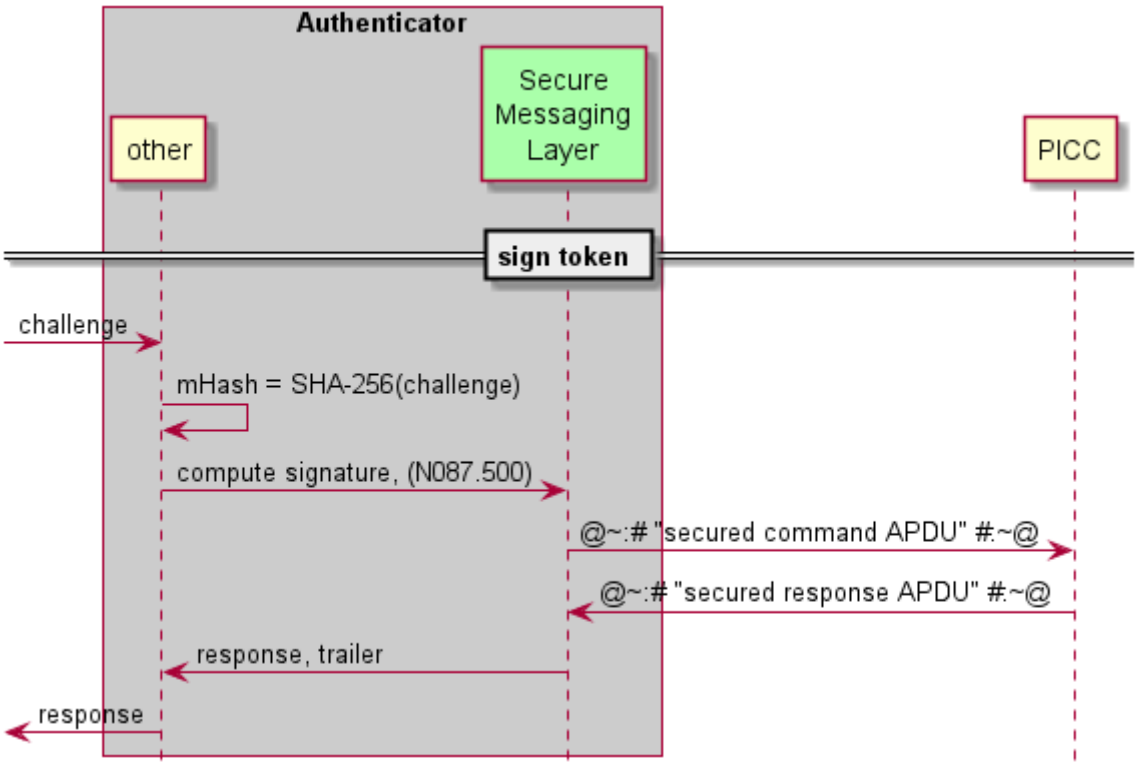

Abbildung 7: *Ablauf eines Signaturvorgangs*

Für den Signaturvorgang ist nur eine Kommando APDU erforderlich: '00 2A 9E9A 20*mHash*00'.

Falls die Antwort APDU Daten enthält (response also nicht leer ist), dann war der Signaturvorgang erfolgreich.