Elektronische Gesundheitskarte und Telematikinfrastruktur

Spezifikation
Identity Provider - Frontend

Version:	1.1.0
Revision:	294968
Stand:	12.11.2020
Status:	freigegeben
Klassifizierung:	öffentlich
Referenzierung:	gemSpec_IDP_Frontend

Dokumentinformationen

Änderungen zur Vorversion

Anpassungen des vorliegenden Dokumentes im Vergleich zur Vorversion können Sie der nachfolgenden Tabelle entnehmen.

Dokumentenhistorie

Version	Stand	Kap./ Seite	Grund der Änderung, besondere Hinweise	Bearbeitung
1.0.0	30.06.20		initiale Erstellung des Dokuments	gematik
1.1.0	12.11.20		Einarbeitung Scope-Themen zu R4.0.1	gematik

Inhaltsverzeichnis

1 Einordnung des Dokumentes

1.1 Zielsetzung

Die vorliegende Spezifikation definiert die Anforderungen zu Herstellung, Test und Betrieb eines Identity Provider (IdP) Clients. Der Client besteht aus den logisch voneinander getrennten Komponenten Anwendungsfrontend und Authenticator-Modul, welche einzeln, aber auch kombiniert in einer Anwendung bereitgestellt werden können. Das Authenticator-Modul übernimmt den Authentifizierungsprozess mit dem IdP-Dienst. Das Anwendungsfrontend ist eine Anwendung, welche Zugriff auf Daten innerhalb der Telematikinfrastruktur (TI) erlangen möchte. Um diesen Zugriff zu erhalten, muss eine Authentifikation über eine Smartcard beim IdP-Dienst durchgeführt werden. Das Authenticator-Modul realisiert dabei den Authentifizierungsprozess und die Kommunikation mit der Smartcard, wodurch das Anwendungsfrontend funktional entlastet wird.

Dieses Dokument beschreibt die normativen Anforderungen sowohl zum Anwendungsfrontend als auch zum Authenticator-Modul. Zudem enthält dieses Dokument informative Hinweise, um bei der Umsetzung zu unterstützen.

1.2 Zielgruppe

Das Dokument richtet sich an Hersteller der Anwendung, welche das Authenticator-Modul und das Anwendungsfrontend beinhaltet. Die Anwendung ist auf Endgeräten des Nutzers (Smartphone oder PC) installiert.

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 diesem Dokument die von Authenticator-Modul und Anwendungsfrontend bereitgestellten Schnittstellen. Benutzte Schnittstellen werden hingegen in der Spezifikation desjenigen Produkttypen beschrieben, der diese Schnittstelle bereitstellt. Auf die entsprechenden Dokumente wird referenziert (siehe auch Anhang 10).

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

Die vollständige Anforderungslage für die in diesem Dokument beschriebenen Anwendungen ergeben sich aus weiteren Konzept- und Spezifikationsdokumenten. Diese sind in dem Produkttypsteckbrief des Produkttyps IdP-Dienst [gemProdT_IDP-Dienst_PTV] verzeichnet.

Nicht Bestandteil des vorliegenden Dokumentes sind Festlegungen zu den Themenbereichen, wie beispielsweise der Umgang des Anwendungsfrontends mit den erlangten Fachdaten, welche proprietären Schnittstellen hierbei verwendet werden oder wie An- oder Abmeldeprozesse außerhalb der OpenID Connect (OIDC)- bzw. OpenAuthorization 2.0 (OAuth2)-Funktionalitäten 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 des Authenticator-Moduls bzw. des Anwendungsfrontends unterscheidet sich vom Systemüberblick des Fachdienstes und des IdP-Dienstes geringfügig, weshalb an dieser Stelle auf die Beschreibung in [gemSpec_IDP_Dienst#Kap. 2] verwiesen wird. Der Unterschied ist, dass es sich beim Primärsystem um eine, aber auch zwei getrennte Anwendungsteile handeln kann, die in dem hier in diesem Dokument beschriebenen Frontend in einer Anwendung zusammengefasst sind. Es findet im Frontend des Versicherten keine Trennung von Anwendungsfrontend und Authenticator-Modul statt.

3 Systemkontext

Das Frontend des Nutzers besteht bei mobiler Nutzung in Form eines mobilen Endgeräts mit zwei logisch voneinander getrennten Komponenten, welche kombiniert in einer Anwendung auf derselben Hardware betrieben werden. Hierbei handelt es sich einerseits um das Authenticator-Modul, welches als dezentraler Teil des IdP-Dienstes den Authentifizierungsprozess wiederverwendbar (für weitere Anwendungsfrontends) kapselt und andererseits um das Anwendungsfrontend, welches Zugriff auf Fachdaten eines Fachdienstes erlangen will. Wenn das Anwendungsfrontend auf einen Fachdienst zugreifen will, muss sich der Nutzer über das Authenticator-Modul beim IdP-Dienst identifizieren. Als Identifikationsnachweis wird vom IdP-Dienst der Besitz der Smartcard (HBA, eGK oder SMC-B) erwartet und zudem, dass der aktuelle Nutzer der Smartcard auch die Kenntnis der dazugehörigen PIN hat.

Die folgende Abbildung skizziert den Systemkontext aus der Sicht eines Endgerätes, auf dem das Authenticator-Modul und das Anwendungsfrontend kombiniert in einer Applikation installiert sind. Daraus ergeben sich Schnittstellen zu dem IdP-Dienst, dem Fachdienst und der Smartcard des Nutzers. Anforderungen zu den Schnittstellen des IdP-Dienstes werden für das Authenticator-Modul in 6.2- Schnittstellen des Authenticator-Moduls beschrieben. Anforderungen zu den Schnittstellen des IdP-Dienstes und des Fachdienstes werden für das Anwendungsfrontend in Kapitel 7.2- Schnittstellen des Anwendungsfrontends beschrieben. Weiterführende Informationen zur Interaktion werden in Kapitel 9- Anhang A – Interaktionen mit Smartcards der TI skizziert.

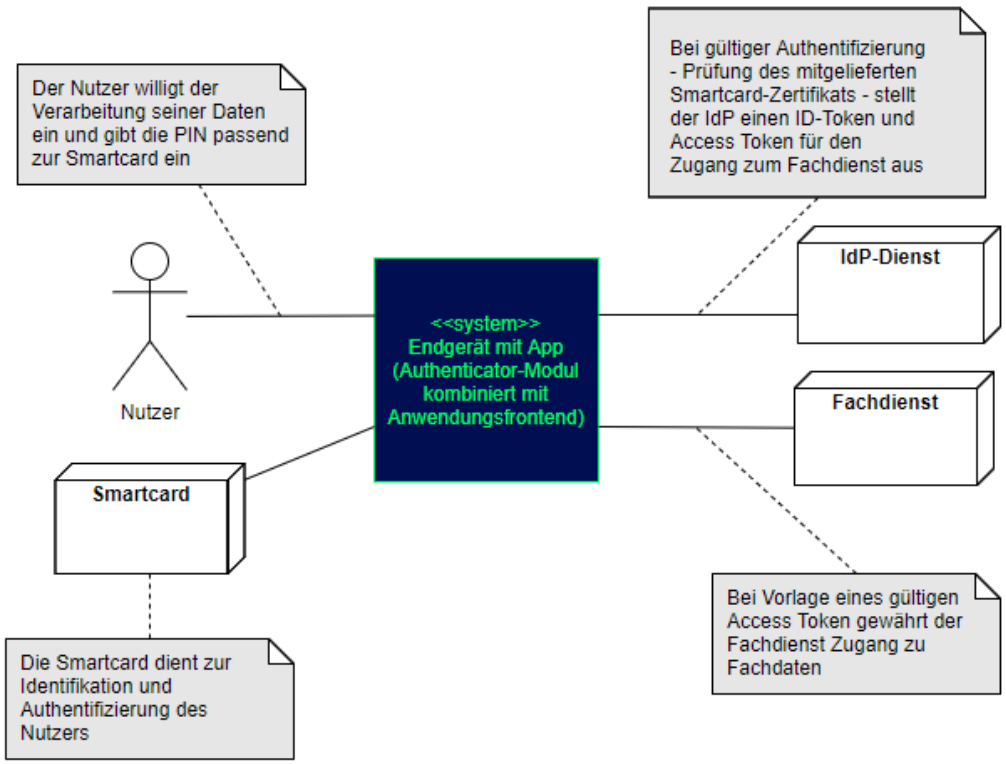

Abbildung 1: Systemkontext aus Sicht des Frontends (bestehend aus Authenticator-Modul und Anwendungsfrontend)

Will ein Nutzer mit seinem Anwendungsfrontend auf einen Fachdienst zugreifen, kann dieser Zugriff nicht direkt erfolgen. Das Authenticator-Modul übernimmt den Aufruf und schickt diesen an den Authorization-Endpunkt. Der Authentifizierungsprozess des Nutzers verläuft zwischen dem Authenticator-Modul und dem IdP-Dienst. Bei einem positiven Verlauf der Authentifizierung liefert das Authenticator-Modul einen Authorization-Code an das Anwendungsfrontend. Gegen Vorlage dieses Codes erhält das Anwendungsfrontend vom IdP-Dienst sowohl einen ID-Token als auch einen `"ACCESS_TOKEN"`. Das Anwendungsfrontend liefert den `"ACCESS_TOKEN"` an den Fachdienst und erhält bei positiver Validierung des Tokens Zugang zu den Fachdienstdaten.

Die Beschreibung der einzelnen Prozessschritte ist im Dokument [gemSpec_IDP_Dienst#Kap.3.3] enthalten.

3.1 Akteure und Rollen

Die Beschreibung der einzelnen Akteure und Rollen ist im Dokument [gemSpec_IDP_Dienst#Kap.3.1] enthalten.

3.2 Nachbarsysteme

Als Nachbarsysteme des Anwendungsfrontends sind das Authenticator-Modul (nur logisch innerhalb einer Anwendung vom Anwendungsfrontend getrennt), der IdP-Dienst (siehe [gemSpec_IDP_Dienst]) sowie die mit dem IdP-Dienst in Verbindung stehenden Fachdienste (siehe [gemSpec_IDP_FD]) zu nennen. Des Weiteren können das Authenticator-Modul und das Anwendungsfrontend als Teil eines Primärsystems realisiert werden. Die Nachbarsysteme ändern sich dadurch nicht.

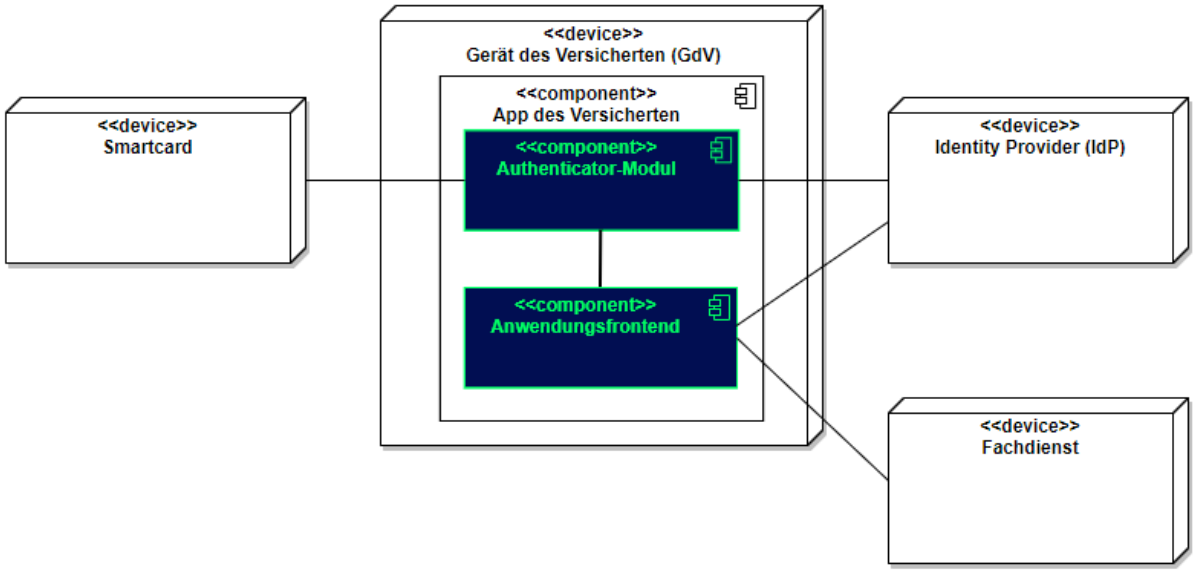

Abbildung 2: Systemüberblick mit Nachbarsystemen

4 Zerlegung des Produkttyps

Das Frontend lässt sich in zwei Module aufteilen: in ein Authenticator-Modul und ein Anwendungsfrontend (GUI), welche in einer Applikation kombiniert sind. Bei der Nutzung innerhalb eines Primärsystems kann das Primärsystem auch vollständig die Funktionalität des Authenticator-Moduls übernehmen. Die Aufteilung in unterschiedliche Module existiert nur logisch, aber nicht in Form von unterschiedlichen Anwendungen.

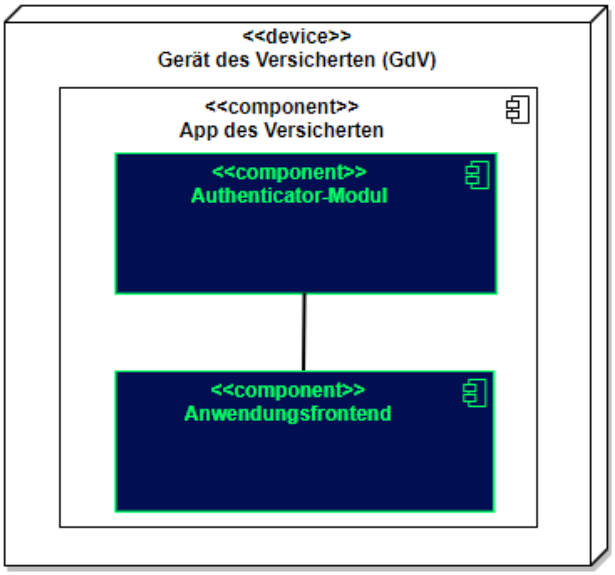

Abbildung 3: Zerlegung des Frontends

Authenticator-Modul

Das Authenticator-Modul bietet die Schnittstelle zum IdP-Dienst an und ist gemeinsam mit dem Anwendungsfrontend in einer mobilen App kombiniert. Für Primärsysteme muss das Authenticator-Modul als Bestandteil des Primärsystems implementiert werden (siehe [gemILF_PS_eRp]). Als Primärsysteme sollen hier PVS (ärztliche und zahnärztliche Praxisverwaltungssystem), KIS (Krankenhausinformationssystem) und AVS (Apothekenverwaltungssystem) genannt sein. Die Beschreibung des Authenticator-Moduls erfolgt in diesem Dokument, weil das Authenticator-Modul einen wesentlichen Bestandteil des Nutzer-Endegrätes/Gerät des Versicherten (GdV) darstellt und somit nicht in der zentralen Providerzone der Telematikinfrastruktur betrieben wird. Authenticator-Modul und Anwendungsfrontend werden in diesem Zusammenhang als ortsveränderliche Komponenten auf unsicheren Endgeräten betrachtet.

Anwendungsfrontend

Das Anwendungsfrontend ist eine Software, welche innerhalb der Telematikinfrastruktur lesend oder schreibend auf die Daten der Fachdienste zugreift.

5 Übergreifende Festlegungen

5.1 Kommunikation mit Diensten der TI

Anwendungsfrontends und das Authenticator-Modul nutzen TLS-Verbindungen für die Kommunikation zu den Diensten der TI.
Das Anwendungsfrontend und das Authenticator-Modul ermitteln die Informationen zu den Endpunkten des Identity Providers aus dem Discovery Document.

A_20606 - Anwendungsfrontend: Kommunikation über TLS-Verbindung
Das Anwendungsfrontend MUSS mit dem IdP-Dienst über TLS kommunizieren.**[<=]**

A_20607 - Authenticator-Modul: Kommunikation über TLS-Verbindung
Das Authenticator-Modul MUSS mit dem IdP-Dienst der TIüber TLS kommunizieren.**[<=]**

A_20608 - Anwendungsfrontend: Unzulässige TLS-Verbindungen ablehnen
Das Anwendungsfrontend MUSS bei jedem Verbindungsaufbau den IdP-Dienst anhand seines TLS-Zertifikats authentifizieren und MUSS die Verbindung ablehnen, falls die Authentifizierung fehlschlägt.**[<=]**

A_20609 - Authenticator-Modul: Unzulässige TLS-Verbindungen ablehnen
Das Authenticator-Modul MUSS bei jedem Verbindungsaufbau den IdP-Dienst anhand seines TLS-Zertifikats authentifizieren und MUSS die Verbindung ablehnen, falls die Authentifizierung fehlschlägt.**[<=]**

A_20610 - Authenticator-Modul: HTTP-Header user-agent
Das Authenticator-Modul MUSS in allen HTTP-Requests an den IdP-Dienst den HTTP-Header „user-agent" gemäß [RFC7231] mit `<Hersteller-ID> <Produktkürzel>/<Produktversion>` gemäß der Produktidentifikation des E-Rezept-FdV befüllen. **[<=]**

A_20612 - Authenticator-Modul: Anzeige bei Ablehnung des Authenticator-Moduls
Das Authenticator-Modul MUSS die Fehlermerldung des IdP-Dienstes an den Benutzer durchreichen, wenn dieser die Durchführung des Authentifizierungsprozesses ablehnt. **[<=]**

6 Funktionsmerkmale Authenticator-Modul

Das Authenticator-Modul ist ein Modul, welches gemeinsam mit dem Anwendungsfrontend in einer Applikation für mobile Endgeräte wie Smartphones bereitgestellt wird. Bei Nutzung eines Primärsystems wird die Funktionalität des Authenticator-Moduls vom Primärsystem selbst realisiert.

Die Bereitstellung der Anwenderfrontends erfolgt 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 iOS findet die sichere Bereitstellung im Apple App Store statt, wobei dem Nutzer auch hier keine Kosten durch den Abruf der Software entstehen dürfen.

Aufgabe des Authenticator-Moduls ist, die von einem Anwendungsfrontend zum Zugriff auf Fachdienste benötigten "ID_TOKEN", "ACCESS_TOKEN" und "SSO_TOKEN" mit Zustimmung des Nutzers (Resource Owner) und nach eingehender Überprüfung dessen Identität am Authorization-Endpunkt zu beantragen. Hierfür wird vom Authorization-Endpunkt ein "AUTHORIZATION_CODE" ausgestellt, der vom Authenticator-Modul an das Anwendungsfrontend übergeben wird. Das gleichzeitig vom Authorization-Endpunkt übergebene "SSO_TOKEN" wird vom Authenticator-Modul selbst gespeichert und wird von diesem für einen zukünftigen Authentifizierungsprozess ohne erneute Abfrage der Zugangsdaten des Nutzers verwendet. Durch Übergabe des "AUTHORIZATION_CODE" erhält das Anwendungsfrontend am Token-Endpunkt das"ID_TOKEN" und "ACCESS_TOKEN".

Die für die Beantragung des "ID_TOKEN" und "ACCESS_TOKEN" notwendigen Informationen bekommt das Authenticator-Modul vom Anwendungsfrontend übergeben. Weitere Informationen bezieht das Authenticator-Modul mittels Near Field Communication-Schnittstelle (NFC) 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. Im Fall eines Primärsystems erfolgt diese Aktion ohne Interaktion mit dem Nutzer im Hintergrund. Weitere nicht normative Informationen hierzu finden sich im Kapitel 9.

6.1 Vorbereitende Maßnahmen

A_20613 - Authenticator-Modul: Regelmäßiges Einlesen des Discovery Document
D as Authenticator-Modul MUSS das Discovery Document [RFC8414] bei eingeschaltetem Gerät regelmäßig alle 24 Stunden einlesen und auswerten, und danach die darin aufgeführten URI zu den benötigten öffentlichen Schlüsseln (Public Keys – PUK) und Diensten verwenden.**[**<=**]**

A_20614 - Authenticator-Modul: Prüfung der Signatur des Discovery Document
Das Authenticator-Modul MUSS die Signatur des Discovery Document mathematisch prüfen und auf ein zeitlich gültiges C.FD.SIG-Zertifikat mit der Rollen-OID *"oid_idpd"* zurückführen können, welches von einer ihm bekannten Komponenten-PKI ausgestellt wurde.**[**<=**]**

Details zur Aktualisierung und sicheren Aufbewahrung der Komponenten-CAs finden sich in 6.6- Zertifikate der Komponenten-PKI.

Details zur Kodierung der Signatur des Discovery Document finden sich in [gemSpec_IDP_Dienst#Kapitel 5.1.3].

6.2 Schnittstellen des Authenticator-Moduls

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

Das Authenticator-Modul nimmt die Authentifizierungs-Anfrage des Anwendungsfrontends entgegen und nutzt den Authorization-Endpunkt des IdP-Dienstes, um die Anfrage einzureichen. Der Authorization-Endpunkt des IdP-Dienstes antwortet – nach positiver Validierung der Anfrage – mit einem "AUTHORIZATION_CODE". Das Authenticator-Modul nimmt den "AUTHORIZATION_CODE" und leitet diesen an das Anwendungsfrontend weiter. Nachfolgende Abbildung skizziert die Schnittstellen des Authenticator-Moduls. Komponenten und Schnittstellen, welche nicht direkt vom Authenticator-Modul genutzt werden, sind in der Abbildung grau hinterlegt.

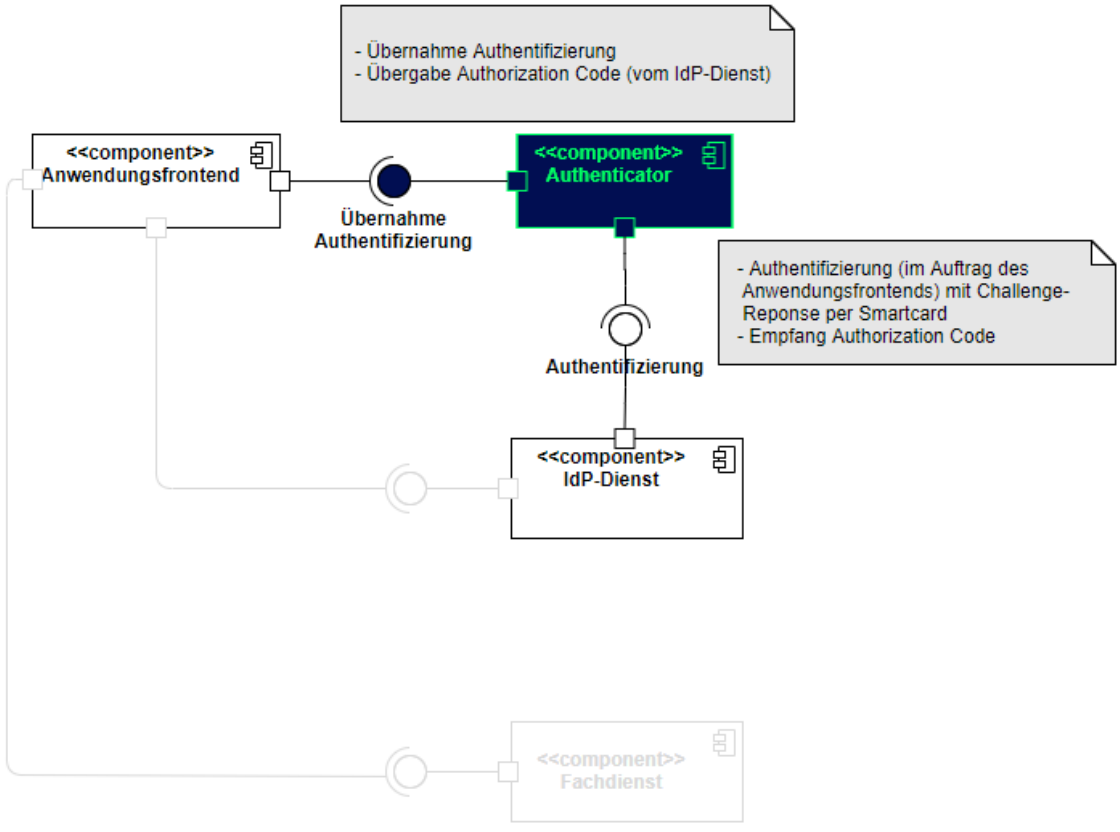

Abbildung 4: Schnittstellen des Authenticator-Moduls

6.2.1 Schnittstellendefinition

Eingehende Daten am Authenticator-Modul stammen vom IdP-Dienst und wurden dort zuvor mit dem aktuellen Signaturzertifikat signiert. Zusätzlich wird die Integrität durch den TLS-Kanal geschützt.

A_20917 - Prüfung auf Vorhandensein und Verwenden eines gültigen "SSO_TOKEN"
Das Authenticator-Modul MUSS vor der Übertragung des Authorization Request (siehe [6.2.1-2- Authenticator-Modul: Übergabe des Authorization-Request an den Authorization-Endpunkt] in diesem Dokument) überprüfen, ob noch ein gültiges `"SSO_TOKEN"` im Dateisystem (siehe [6.2.1-10- Authenticator-Modul: Temporäre Speicherung von "SSO_TOKEN"] in diesem Dokument) vorhanden ist.
Ist ein solches `"SSO_TOKEN"` im Dateisystem vorhanden, MUSS das Authenticator-Modul dieses `"SSO_TOKEN"` an den Authorization-Endpunkt senden.
Ist kein solches `"SSO_TOKEN"` im Dateisystem vorhanden, MUSS das Authenticator-Modul die Anfrage gemäß [6.2.1-2- Authenticator-Modul: Übergabe des Authorization-Request an den Authorization-Endpunkt] fortsetzen und somit einen neuen Authentifizierungsprozess einleiten.**[<=]**

Hinweis: Die Prüfung der Gültigkeit des `"SSO_TOKEN"` ergibt sich aus dessen maximaler Gültigkeitsdauer, welche je nach Fachdienst unterschiedlich sein kann und insgesamt durch den Anbieter des IDP-Dienstes eingegrenzt ist. Eine Integritätsprüfung des `"SSO_TOKEN"` ist nicht möglich, weil das `"SSO_TOKEN"` für den Authorization-Endpunkt mit dessen öffentlichen Schlüssel verschlüsselt ist.

A_20601 - Authenticator-Modul: Übergabe des Authorization-Request an den Authorization-Endpunkt
Das Authenticator-Modul MUSS den Authorization-Request, welchen dieses vom Anwendungsfrontend erhalten hat, an den Authorization-Server des IdP-Dienstes schicken. Der Authorization-Request MUSS folgende Parameter enthalten:

- `"response_type"`
- `"scope"`
- `"client_id"`
- `"redirect_uri"`
- `"code_challenge"` (Hashwert des `"code_verifier"`) [RFC7636 # section-4.2]
- `"code_challenge_method"` HASH-Algorithmus (S256) [RFC7636 # section-4.3]

[<=]

Hinweis: Der folgende Aufruf skizziert einen beispielhaften HTTP-GET-Request an den IdP-Dienst, welcher vom Authenticator-Modul initiiert wird:

```
GET /auth?response_type=code&scope=openid%20e-
rezept&state=af0ifjsldkj&client_id=ZXJlemVwdC1hcHA&redirect_uri=https%3A%2F
%2Fapp.e-
rezept.com%2Fauthnres&code_challenge_method=S256&code_challenge=S41HgHxhXL1
CIpfGvivWYpbO9b_QKzva-9ImuZbt0Is

HTTP/1.1
Host: idp.com
X-Authenticator-Modul: 1.0
```

```
Accept: application/json
User-Agent: Authenticator-Modul/1.0
```

A_20600 - Authenticator-Modul: Annahme des "user_consent" und des "CHALLENGE_TOKEN"
Das Authenticator-Modul MUSS den "user_consent" und den "CHALLENGE_TOKEN" vom Authorization-Endpunkt des IdP-Dienstes entgegennehmen.**[<=]**

Hinweis: Der Authorization-Endpunkt des IdP-Dienstes, welcher die Nutzerauthentifizierung durchführt und für die Ausstellung des "AUTHORIZATION_CODE" zuständig ist, liefert den "user_consent" und das "CHALLENGE_TOKEN" als Antwort auf den Authorization-Request des Authenticator-Moduls.

Nachfolgend wird beispielhaft ein "CHALLENGE_TOKEN" in Form eines JSON Web Token (JWT) dargestellt:

```
Challenge JWT:
challenge_headers = {
    "typ":"JOSE+JSON",
    "iat":1591714252326,
    "exp":1591714552326,
    "jti":"c3a8f9c8-aa62-11ea-ac15-6b7a3355d0f6",
    "snc":"sLlxlkskAyuzdDOwe8nZeeQVFBWgscNkRcpgHmKidFc"
}
challenge_payload = {
    "response_type":"code",
    "scope":"openid e-rezept",
    "client_id":"ZXJlemVwdC1hcHA",
"state":"af0ifjsldkj",
    "redirect_uri":"https://app.e-rezept.com/authnres",
    "code_challenge_method":"S256",
    "code_challenge":"S41HgHxhXL1CIpfGvivWYpbO9b_QKzva-9ImuZbt0Is"
}
```

Der Authorization-Endpunkt des IdP-Dienstes hat den "CHALLENGE_TOKEN" mit seinem privaten Schlüssel "PRK_AUTH" signiert. Der folgende Aufruf skizziert beispielhaft die Antwort des Authorization-Endpunktes, welche vom Authenticator-Modul angenommen wird. Der "CHALLENGE_TOKEN" wird dabei nur angedeutet:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "challenge":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpPU0UrSlNPTiIsImlhdCI6MTU5MTcxNDI1MjMy......
",
  "user_consent": {
    "client_name": "E-Rezept App",
    "url": "https://e-rezept.com/",
    "requested_scope": {
      "openid": "Der Zugriff auf den ID-Token"
      "e-rezept": "Zugriff auf die E-Rezept-Funktionalität."
    },
"show_once": true,
"amr": ["JWT-Challenge-Response"],
```

```
"name": "Zustimmung zur Verarbeitung des Namens des Versicherten",
"idNummer": "Zustimmung zur Verarbeitung der Krankenversichertennummer"
    // ggf. mehr Informationen, welche dem Nutzer angezeigt werden sollen,
wie die Auflistung der mit der Zustimmung weitergegebenen Daten
  }
}
```

A_20525 - Authenticator-Modul: Anzeige des "user_consent" und PIN-Abfrage
Das Authenticator-Modul MUSS im Zusammenhang mit der PIN-Abfrage für die Signatur des "CHALLENGE_TOKEN" durch die Smartcard im selben Dialog die Consent-Freigabe des "user_consent" durch den Nutzer einfordern, damit dieser durch die PIN-Eingabe seine Willenserklärung abgibt und der Verwendung seiner Daten in diesen Claims zustimmt.**[<=]**

Hinweis: Bei Primärsystemen kann auf eine erneute PIN-Eingabe und Consent-Freigabe verzichtet werden, solange sich die SMC-B im freigeschalteten Modus befindet.

A_19908-01 - Authenticator-Modul: Prüfung der Signatur des "CHALLENGE_TOKEN"
Das Authenticator-Modul MUSS die Signatur des "CHALLENGE_TOKEN" gegen den aktuellen öffentlichen Schlüssel des Authorization-Endpunktes "PUK_AUTH" prüfen.Liegt dem Authenticator-Modul der öffentliche Schlüssel des Authorization-Endpunktes noch nicht vor,MUSS es diesen vom Authorization Server gemäß den Angaben der Adresse PUK_URI_AUTH im Discovery Document abrufen.**[<=]**

Das Authenticator-Modul wird vom Anwendungsfrontend zur Authentifizierung gegenüber dem IdP-Dienst herangezogen. Das Anwendungsfrontend ist eine beim IdP-Dienst als "OpenID Connect-Client" registrierte Software. Das Anwendungsfrontend erhält seinerseits bei der Registrierung am IdP-Dienst einen eindeutigen Identifier.

Da sich Authenticator-Modul und Anwendungsfrontend immer auf ein und demselben Endgerät des Nutzers befinden, ist es nicht notwendig, dass der IdP-Dienst deren Adressierung im Voraus kennt. Will nun das Anwendungsfrontend einen Zugriff auf den entsprechenden Fachdienst initiieren, leitet es eine Anfrage an die vom IdP-Dienst bekanntgegebene URI. Die Anfrage wird innerhalb der Anwendung dem Authenticator-Modul zugewiesen. Das Authenticator-Modul bereitet die Anfrage auf und prüft, ob alle Voraussetzungen erfüllt sind. Dann leitet das Authenticator-Modul die Anfrage an den Authorization-Endpunkt des IdP-Dienstes weiter. Der IdP-Dienst erzeugt eine "session_id" und fordert (im Falle eines mobilen Endgerätes) vom Authenticator-Modul die Signatur einer "CHALLENGE" durch das vorgesehene Identifikationsmittel. Im Falle eines Versicherten die eGK, und im Falle einer Leistungserbringerinstitution die SM-B welche über das Primärsystems durch die Operation ExternalAuthenticate des Konnektors gemäß [gemSpec_Kon#4.1.13.4] bzw. [gemILF_PS#4.4.6.1] angesprochen wird. Das Token wird hierbei nicht selbst signiert, sondern der darüber gebildete HASH-Wert (siehe Abschnitt9.3.7.3- Signiervorgang in diesem Dokument).

A_20700-04 - Authenticator-Modul: Signatur der "CHALLENGE"
Das Authenticator-Modul MUSS die vom Authorization-Endpunkt empfangene "CHALLENGE"mit dem aus der Smartcard des Nutzers empfangenen Zertifikat C.CH.AUT gemäß [gemSpec_Krypt#5.6.2 ECDSA-Signaturen im CMS-Format] signieren. Hierbei wird der über die "CHALLENGE" gebildete HASH-Wert zur Signatur überreicht (siehe Abschnitt 9.3.7.3- Signiervorgang in diesem Dokument).**[<=]**

A_20526 - Authenticator-Modul: Response auf die "CHALLENGE" des Authorization-Endpunktes

Das Authenticator-Modul MUSS:
- das eingereichte "CHALLENGE_TOKEN",
- die von der Smartcard signierten Challenge-Signatur "challenge_sig" und
- das Authentifizierungszertifikat der verwendeten Smartcard – mit dem öffentlichen Schlüssel des Authorization-Endpunktes "PUK_AUTH" verschlüsselt –
zusammen in Form eines HTTP-POST-Requests an den Authorization-Endpunktes senden. **[<=]**

Hinweis: Das Signieren und Verschlüsseln des "CHALLENGE_TOKEN" ist durch die Verwendung eines Nested JWT [RFC7519 # appendix-A.2] zu realisieren. Das Signieren wird dabei durch die Verwendung einer JSON Web Signature (JWS) [RFC7515 # section-3] gewährleistet. Die Verschlüsselung des signierten Tokens wird durch die Nutzung der JSON Web Encryption (JWE) [RFC7516 # section-3] sichergestellt.

Der folgende beispielhafte Aufruf skizziert den HTTP-POST-Request, welcher vom Authenticator-Modul an den Authorization-Endpunkt des IdP-Dienst übertragen wird. Dabei wird das signierte und verschlüsselte "CHALLENGE_TOKEN" nur angedeutet:

```
POST /sign_response HTTP/1.1
Host: idp.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Anwendungsfrontend-App/1.0
signed_challenge=eyJhbGciOiJFUzI1NiIsInR5cCI6IkpPU0UrSlNPTiIsIng.....
```

A_20527 - Authenticator-Modul: Übertragung des "AUTHORIZATION_CODE" an das Anwendungsfrontend
Das Authenticator-Modul MUSS den vom Authorization-Endpunkt empfangenen "AUTHORIZATION_CODE" an das Anwendungsfrontend übertragen.**[<=]**

Hinweis: Der Authorization-Endpunkt liefert den "AUTHORIZATION_CODE" gemeinsam mit dem "SSO_TOKEN " innerhalb einer HTTP-Redirection (HTTP-Status Code 302) an das Authenticator-Modul zurück. Der Wert des Attributs "location" der HTTP 302 Response ist die vom Anwendungsfrontend beim mobilen Betriebssystem registrierte URI. Beim Aufruf der URI wird automatisch das Anwendungsfrontend mit der Verarbeitung der URI gestartet.

Nachfolgend wird ein beispielhafter Response des IdP skizziert, welcher vom Authenticator-Modul an das Anwendungsfrontend weitergereicht wird, dabei werden sowohl der "AUTHORIZATION_CODE", als auch der "SSO_TOKEN" nur angedeutet:

```
HTTP/1.1 302 Found

User-Agent: Anwendungsfrontend-App/1.0
Location: https://app.e-rezept.com/authnres?code=eyJhbGciOiJkaXIiL...
&ssotoken=eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwiZXhwIjoxNTkxNzE0NjU....
&state=af0ifjsldkj
```

A_20284 - Authenticator-Modul: Annahme von "SSO_TOKEN"
Das Authenticator-Modul MUSS das vom Token-Endpunkt ausgegebene "SSO_TOKEN" in der HTTP/1.1 Statusmeldung 302 verarbeiten. Das Authenticator-Modul MUSS das "SSO_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird.**[<=]**

Hinweis: Der IdP-Dienst hat das "SSO_TOKEN" mit dem privaten Schlüssel "PRK_AUTH" für sich signiert und verschlüsselt. Das Authenticator-Modul kann und braucht den Inhalt des Tokens nicht zu lesen. Durch die Vorlage des "SSO_TOKEN" kann bei Bedarf ein neuer

"AUTHORIZATION_CODE" am Authorization-Endpunkt ohne die erneute Authentisierung des Nutzers erzeugt werden.

A_20499 - Authenticator-Modul: Temporäre Speicherung von "SSO_TOKEN"
Das Authenticator-Modul MUSS beim aktiven Beenden der Anwendung vorhandene "SSO_TOKEN" aus dem RAM sowie lokal gespeicherte Kopien desselben sicher löschen.**[**<=**]**

6.2.2 Nutzung

Die im Abschnitt 6.2.1 beschriebenen Schnittstellen des Authenticator-Moduls werden durch das Anwendungsfrontend genutzt. Die Nutzung durch das Anwendungsfrontend erfolgt hierbei zwangsläufig vom gleichen Gerät aus.

Die verwendeten Standards sind:

- RFC3986 (URI)
- 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 (OAuth 2.0 Authorization)
- RFC6750 (OAuth 2.0 Bearer)
- RFC7636 (Proof Key for Code Exchange by OAuth Public Clients)
- RFC7662 (OAuth 2.0 TOKEN INTROSPECTION)
- RFC8252 (OAuth 2.0 for Native Apps)

6.3 Hardwaremerkmale

Das Authenticator-Modul greift auf die NFC-Schnittstelle des Nutzer-Endgerätes oder auf 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 über NFC mit der eGK oder dem HBA zu kommunizieren. Die hierfür notwendige Middleware ist als gegebene Voraussetzung anzusehen und somit Bestandteil des Betriebssystems bzw. wird zusammen mit dem Authenticator-Modul installiert und ist insofern durch den Anbieter des IdP-Dienstes bereitzustellen.

6.4 Zertifikatsprüfung

Das Authenticator-Modul verwendet bei den in TAB_Authenticator_001 dargestellten Aktivitäten Zertifikate.

Tabelle 1: TAB_Authenticator_001 – Zertifikatsnutzung des Authenticator-Modul

Aktivität	Zertifikat der TI	Zertifikatstyp	Rollen-OID	Nutzung
TLS-Verbindungsaufbau zum IdP-Dienst	nein	TLS Internet Zertifikat	n/a	aktiv
Prüfung des Discovery Document	ja	C.FD.SIG	oid_idpd	aktiv
Signatur der Challenge des IdP mittels Smartcard	ja	C.CH.AUT C.HP.AUT C.HCI.AUT	oid_versicherter gemäß gemSpec_OID#Tab_PKI_402 gemäß gemSpec_OID#Tab_PKI_403	passiv

A_20617-01 - Authenticator-Modul: Verpflichtende Zertifikatsprüfung
Das Authenticator-Modul MUSS aktiv verwendete Zertifikate (bspw. für den TLS-Verbindungsaufbau), welche auf Root-Zertifikaten aus der TSL basieren, gemäß "TUC_PKI_018" auf Integrität und Authentizität prüfen. Das Authenticator-Modul MUSS die von dem Zertifikat und den darin enthaltenen Attributen (bspw. öffentliche Schlüssel) abhängenden Arbeitsabläufe ablehnen, wenn die Prüfung kein positives Ergebnis ("gültig") liefert. Das Authenticator-Modul MUSS alle öffentlichen Schlüssel, die es verwenden will, auf eine positiv verlaufene Zertifikatsprüfung zurückführen können.**[<=]**

Hinweis: "Ein Zertifikat aktiv verwenden" bedeutet im Sinne von [6.4-1- Authenticator-Modul: Verpflichtende Zertifikatsprüfung], dass ein Authenticator-Modul einen dort aufgeführten öffentlichen Schlüssel innerhalb einer kryptografischen Operation (Signaturprüfung, Verschlüsselung, Signaturprüfung von öffentlichen (EC)DH-Schlüsseln etc.) nutzt. Erhält ein Authenticator-Modul bspw. einen Access Token, in dem Signaturen und Zertifikate enthalten sind, und behandelt es diesen Token als opakes Datenobjekt, ohne die Zertifikate darin gesondert zu betrachten, dann verwendet das Authenticator-Modul diese Zertifikate im Sinne von [6.4-1- Authenticator-Modul: Verpflichtende Zertifikatsprüfung] passiv.

6.5 Zertifikatsprüfung von Internet-Zertifikaten

Folgende Vorgaben gelten für die Prüfung von Internet-Zertifikaten.

A_20068-01 - Authenticator-Modul: Prüfung Internet-Zertifikate
Das Authenticator-Modul MUSS das Internet-seitige Zertifikat des IdP-Dienstes prüfen. Hierfür MUSS das Authenticator-Modul sowohl eine Signaturprüfung als auch eine Prüfung der zeitlichen Gültigkeit durchführen. Falls diese Prüfung negativ ausfällt, MUSS es das Zertifikat als "ungültig" bewerten.
Das Authenticator-Modul MUSS das Zertifikat anhand der Signaturprüfung 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, zurückführen können. Ansonsten MUSS es das Zertifikat als "ungültig" bewerten.**[<=]**

Hinweis: Eine positiv ausgefallene Signaturprüfung von A_20068-01 ist gleichbedeutend damit, dass das CA-Zertifikat im Zertifikats-Truststore eines aktuellen Webbrowsers vorhanden ist.

A_20618 - Authenticator-Modul: Unzulässige TLS-Verbindungen ablehnen
Das Authenticator-Modul MUSS bei jedem Verbindungsaufbau den IdP-Dienst anhand seines TLS-Zertifikats authentifizieren und MUSS die Verbindungen ablehnen, falls die Authentifizierung fehlschlägt.**[<=]**

Der IdP-Dienst authentisiert sich mit einem extended-validation-X.509-Zertifikat.
Es gelten die Bedingungen für den TLS-Handshake gemäß [gemSpec_PKI#GS-A_4662].

6.6 Zertifikate der Komponenten-PKI

A_20743 - Prüfung der Aktualität der Komponenten-CA-Zertifikate
Der Anbieter des Authenticator-Moduls MUSS sicherstellen, dass das aktuell verwendete Authenticator-Modul mit den gültigen Zertifikaten der Komponenten-CA der TI arbeitet. Er MUSS ebenfalls hierzu mindestens einmal monatlich einen Abgleich der im Authenticator-Modul eingebundenen Zertifikate mit den in der TSL hinterlegten durchführen. Der Anbieter des Authenticator-Moduls MUSS ein Update veröffentlichen, wenn der Fingerprint der Zertifikate in der TSL nicht mit denen im Authenticator-Modul übereinstimmen.**[<=]**

7 Funktionsmerkmale Anwendungsfrontend

Das Anwendungsfrontend ist eine Software-Komponente, welche darauf zugeschnitten ist, Daten 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 Integrität hin zu überprüfen und gleichzeitig sicherzustellen, dass der vortragende Akteur vermeintlich 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 oder dezentrale Plattformleistung angebotene Fachdienste zugreifen zu können.

7.1 Vorbereitende Maßnahmen

A_20603 - Organisatorische Registrierung des Anwendungsfrontends
Das Anwendungsfrontend MUSS sich über einen organisatorischen Prozess am IdP-Dienst registrieren und die vom IdP-Dienst dabei vergebene "`client_id`" im Anwendungsfrontend speichern. Diese MUSS vom Anwendungsfrontend bei Nutzung des IdP-Dienstes übertragen werden.**[<=]**

A_20740 - Bekanntgabe der Redirect-URI des Anwndungsfrontend
Das Anwendungsfrontend MUSS beim IdP-Dienst bei der Registrierung eine "`redirect_uri`" hinterlegen. **[<=]**

Hinweis: Der IdP-Dienst nutzt die registrierte "`redirect_uri`" im späteren Verlauf dazu, eine Redirection auszuführen. Dabei wird der ausgestellte "`AUTHORIZATION_CODE`" vom Authenticator-Modul an das Anwendungsfrontend weitergeleitet.

A_20741 - Speicherung des Downloadpunktes des Discovery Document im Anwendungsfrontend
Das Anwendungsfrontend MUSS den vom IdP-Dienst bei der Registrierung bekanntgegebenen Downloadpunkt des Discovery Document als konfigurierbaren Parameter speichern. **[<=]**

Hinweis: Über das Discovery Document können u. a. die URIs der Endpunkte des IdP-Dienstes und die Adressen der dazugehörigen öffentlichen Schlüssel bezogen werden.

A_20501 - Einbindung des Primärsystems
Das Primärsystem (PVS, AVS und KIS) MUSS eine Schnittstelle implementieren, welche die Funktionalität des Authenticator-Moduls übernimmt.**[<=]**

Hinweis: Die Aufgabe "Challenge" wird durch den Konnektor mittels Zugriff auf die im Kartenterminal gesteckte und bereits freigeschaltete SMC-B des Leistungserbringers signiert und die Aufgaben-Lösung "Response" über das Primärsystem an den IdP-Dienst zurück übertragen.

A_20512 - Regelmäßiges Einlesen des Discovery Document
Das Anwendungsfrontend MUSS das Discovery Document [RFC8414] löschen, wenn dieses 24 Stunden alt oder älter ist. Das Anwendungsfrontend MUSS das Discovery Document neu herunterladen, einlesen und auswerten und danach die darin aufgeführten URI zu den benötigten öffentlichen Schlüsseln (PUKs) und Diensten verwenden, wenn kein aktuelles Discovery Document vorliegt. **[<=]**

Hinweis: Der IdP-Dienst übergibt den Downloadpunkt während der organisatorischen Registrierung des Anwendungsfrontends bzw. des Primärsystems beim IdP-Dienst.

A_20623 - Anwendungsfrontend: Prüfung der Signatur des Discovery Document
Das Anwendungsfrontend MUSS die Signatur des Discovery Document mathematisch prüfen und auf ein zeitlich gültiges C.FD.SIG-Zertifikat mit der Rollen-OID `"oid_idpd"` zurückführen können, welches rückführbar ist auf ein CA-Zertifikat aus einer authentischen, integren und zeitlich gültigen TSL.**[<=]**

7.2 Schnittstellen des Anwendungsfrontends

Das Anwendungsfrontend hat Schnittstellen zum logisch getrennten, aber in einer Applikation kombinierten Authenticator-Modul, sowie zum IdP-Dienst und zum Fachdienst.

Es initiiert seine Authentifizierungsanfrage, welche vom Authenticator-Modul übernommen und im Auftrag des Anwendungsfrontends beim IdP-Dienst eingereicht wird. Nach erfolgreicher Authentifizierung übergibt der IdP-Dienst einen "`AUTHORIZATION_CODE`" an das Authenticator-Modul zurück. Dieser Code wird an das Anwendungsfrontend weitergereicht. Das Anwendungsfrontend erhält vom IdP-Dienst durch Vorlage des Codes einen "`ID_TOKEN`" und einen "`ACCESS_TOKEN`". Durch Vorlage des "`ACCESS_TOKEN`" erhält das Anwendungsfrontend Zugriff auf die Daten des Fachdienstes.
Nachfolgende Abbildung skizziert die beschriebenen Schnittstellen des Anwendungsfrontends. Schnittstellen zwischen anderen Komponenten sind dabei grau angedeutet.

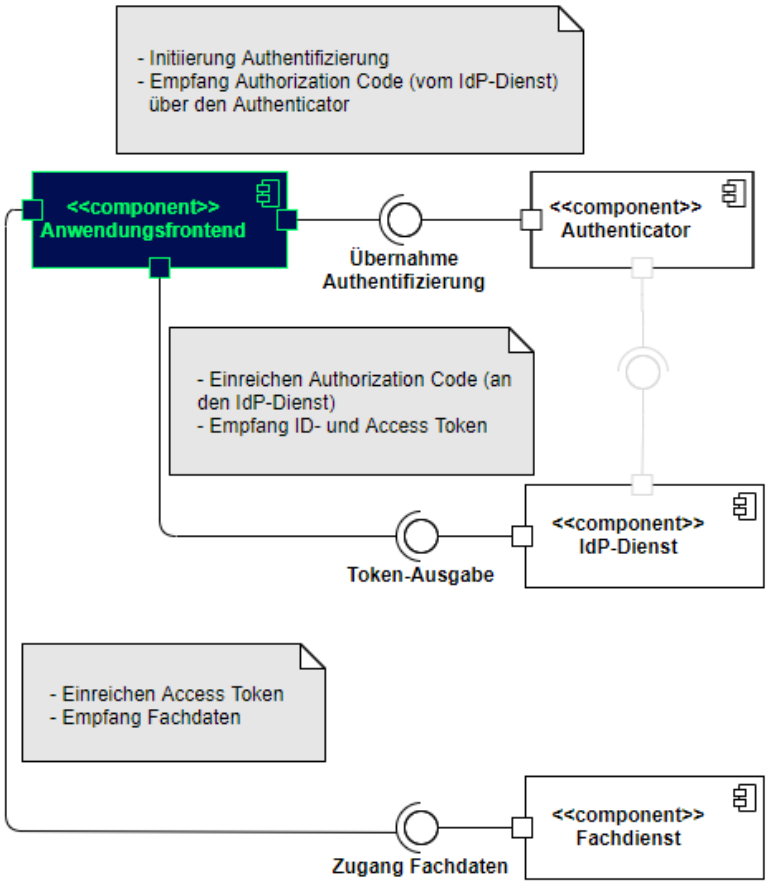

Abbildung 5: Schnittstellen des Anwendungsfrontends

7.2.1 Schnittstellendefinition

Das Anwendungsfrontend übergibt seine Authentifizierungs-Anfrage an das Authenticator-Modul. Es reicht den vom Authenticator-Modul übergebenen "AUTHORIZATION_CODE" beim IdP-Dienst ein und erhält nach positiver Validierung einen "ID_TOKEN" und einen "ACCESS_TOKEN". Das Anwendungsfrontend nutzt den "ACCESS_TOKEN", um Fachdaten vom Fachdienst anzufragen.

A_20309 - Bildung von "CODE_VERIFIER" und "CODE_CHALLENGE"
Das Anwendungsfrontend MUSS zur Laufzeit einen "CODE_VERIFIER" (Zufallswert) gemäß [RFC7636 # section-4.1] bilden. Der "CODE_VERIFIER" MUSS eine Entropie von mindestens 43 und maximal 128 Zeichen enthalten. Das Anwendungsfrontend MUSS über den "CODE_VERIFIER" einen HASH-Wert, die sogenannte "CODE_CHALLENGE", gemäß [RFC7636 # section-4.2] bilden.**[<=]**

A_20483 - Formulierung und Inhalte der Anfrage zum "AUTHORIZATION_CODE" für einen "ACCESS_TOKEN"

Das Anwendungsfrontend MUSS über das Authenticator-Modul den Antrag zum"AUTHORIZATION_CODE" für einen "ACCESS_TOKEN" via Private-Use URI Scheme Redirection [RFC8252 # section-7.1] beim Authorization-Endpunkt in Form eines HTTP/1.1 GET-Request stellen und dabei die folgenden Attribute anführen:

- "response_type"
- "scope"
- "client_id"
- "redirect_uri"
- "code_challenge" (Hashwert des "code_verifier") [RFC7636 # section-4.2]
- "code_challenge_method" HASH-Algorithmus (S256) [RFC7636 # section-4.3]

[<=]

Hinweis: Der folgende Aufruf skizziert einen beispielhaften HTTP-GET-Request an den IdP-Dienst, welcher vom Betriebssystem gemäß [RFC8252] an das Authenticator-Modul umgeleitet und dort schließlich ausgeführt wird:

```
GET /auth?response_type=code&scope=openid%20e-
rezept&state=af0ifjsldkj&client_id=ZXJlemVwdC1hcHA&redirect_uri=https%3A%2F
%2Fapp.e-
rezept.com%2Fauthnres&code_challenge_method=S256&code_challenge=S41HgHxhXL1
CIpfGvivWYpbO9b_QKzva-9ImuZbt0Is

HTTP/1.1
Host: idp.com
X-Anwendungsfrontend-App: 1.0
Accept: application/json
User-Agent: Anwendungsfrontend-App/1.0
```

A_20624 - Anwendungsfrontend: Prüfung der Signatur des AUTHORIZATION_CODE
Das Anwendungsfrontend MUSS die Signatur des AUTHORIZATION_CODE mathematisch prüfen und auf ein zeitlich gültiges C.FD.SIG-Zertifikat mit der Rollen-OID oid_idpd zurückführen können, welches rückführbar ist auf ein CA-Zertifikat aus einer authentischen, integren und zeitlich gültigen TSL.**[<=]**

A_20085 - Fehlermeldungen des Anwendungsfrontends
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_20529 - Senden von "AUTHORIZATION_CODE" und "code_verifier" an den Token-Endpunkt
Das Anwendungsfrontend MUSS "AUTHORIZATION_CODE" und "code_verifier" TLS-gesichert als HTTP/1.1 POST Request an den Token-Endpunkt senden und dort gegen das "ID_TOKEN" und "ACCESS_TOKEN" eintauschen.**[<=]**

Hinweis: Der folgende Aufruf skizziert beispielhaft den HTTP-POST-Request des Anwendungsfrontends an den Token-Endpunkt. Der mitgegebene "AUTHORIZATION_CODE" wird dabei nur angedeutet:

```
POST /token HTTP/1.1
Host: idp.com
Content-Type: application/x-www-form-urlencoded
 User-Agent: Anwendungsfrontend-App/1.0
grant_type=authorization_code
&code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwiZXhwIjoxNTkx.....
&redirect_uri=https%3A%2F%2Fapp.e-rezept.com%2Fauthnres
&code_verifier=MApN61C4itdm4-58dCjMkoucuu00jipPlNibsAxjyJk
```

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

A_20605 - Fehlermeldung des Token-Endpunktes Formatierung
Das Anwendungsfrontend MUSS den Inhalt der Fehlermeldungen sowie mögliche Hinweise zur Fehlervermeidung vom Token-Endpunkt übernehmen.**[<=]**

Hinweis: Es ist insbesondere der Inhalt der Fehlermeldung gemeint. Die Formatierung darf den Gegebenheiten des Endgerätes entsprechend angepasst werden.

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_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.**[<=]**

Hinweis: Das Anwendungsfrontend nimmt sowohl den "ID_TOKEN", als auch den "ACCESS_TOKEN" aus der Antwort des Token-Endpunktes des IdP-Dienstes. Der Token-Endpunkt antwortet mit den Token auf die erfolgreiche Übergabe und Validierung des "AUTHORIZATION_CODE" durch das Anwendungsfrontend. Nachfolgend wird beispielhaft die Antwort des Token-Endpunktes skizziert. Der "ID_TOKEN" und der "ACCESS_TOKEN" werden dabei nur angedeutet:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{"token_type": "Bearer",
 "expires_in": 300,
 "id_token": "...",
 "access_token": "...",

...
}
```

A_20625 - Anwendungsfrontend: Prüfung der Signatur des ID_TOKEN
Das Anwendungsfrontend MUSS die Signatur des "ID_TOKEN" mathematisch prüfen und auf ein zeitlich gültiges C.FD.SIG-Zertifikat mit der Rollen-OID oid_idpd zurückführen können, welches rückführbar ist auf ein CA-Zertifikat aus einer authentischen, integren und zeitlich gültigen TSL.**[<=]**

A_20283 - Annahme des "ACCESS_TOKEN"
Das Anwendungsfrontend MUSS das vom Token-Endpunkt ausgegebene "ACCESS_TOKEN" in der HTTP/1.1 Statusmeldung 200 verarbeiten. Das Anwendungsfrontend MUSS das

"ACCESS_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird.**[**<=**]**

A_20602 - Einreichen des "ACCESS_TOKEN" beim Fachdienst
Das Anwendungsfrontend MUSS das "ACCESS_TOKEN" im Rahmen des entsprechenden fachlichen Aufrufs beim Fachdienst einreichen, um Zugang zu den angeforderten Daten zu erhalten.**[**<=**]**

8 Verteilungssicht

Die Beschreibung der theoretisch möglichen Verwendung unterschiedlicher Authenticator-Modul-Profile bleibt hier aus, da durch das beschriebene 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.

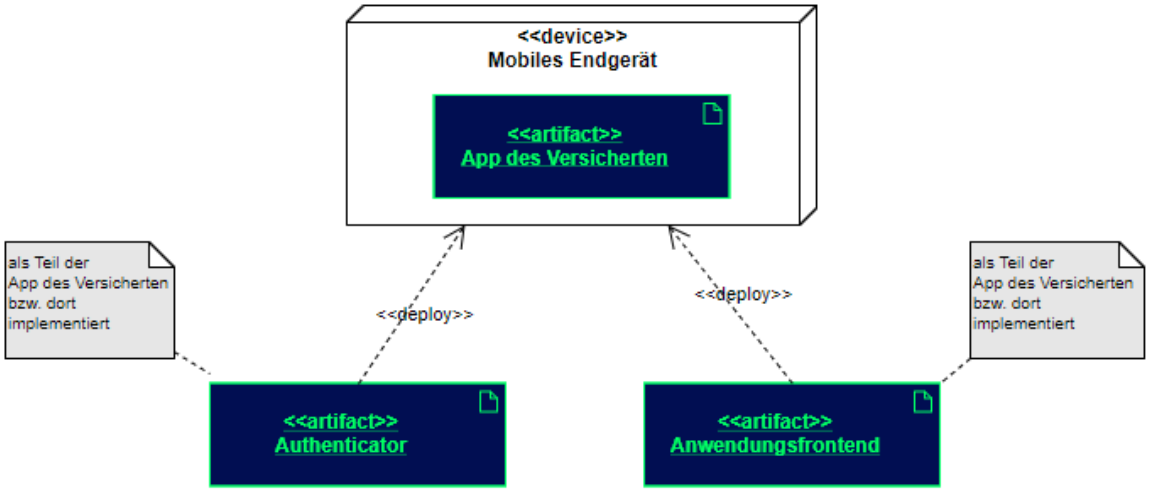

Abbildung 6: Verteilungssicht beim Einsatz eines mobilen Endgerätes

Teilsysteme (Module der App des Versicherten) sind zusammen auf einem Gerät, aber niemals auf mehreren Geräten verteilt, vorhanden. Als Liste der möglichen Teilsysteme seien die folgenden genannt:

- Mobiles Endgerät des Nutzers (z. B. Android Smartphone, Apple iPhone)
- Stationäres Endgerät des Nutzers (z. B. Konnektor, PVS, AVS, KIS)

Hinweis: Die beiden Teilsysteme Authenticator-Modul und Anwendungsfrontend können in zukünftigen Versionen ggf. auch auf getrennten Endgeräten <<device>> betrieben werden.

Bei der Nutzung eines stationären Endgerätes und eines Primärsystems, sind das Authenticator-Modul und das Anwendungsfrontend Teil des Primärsystems.

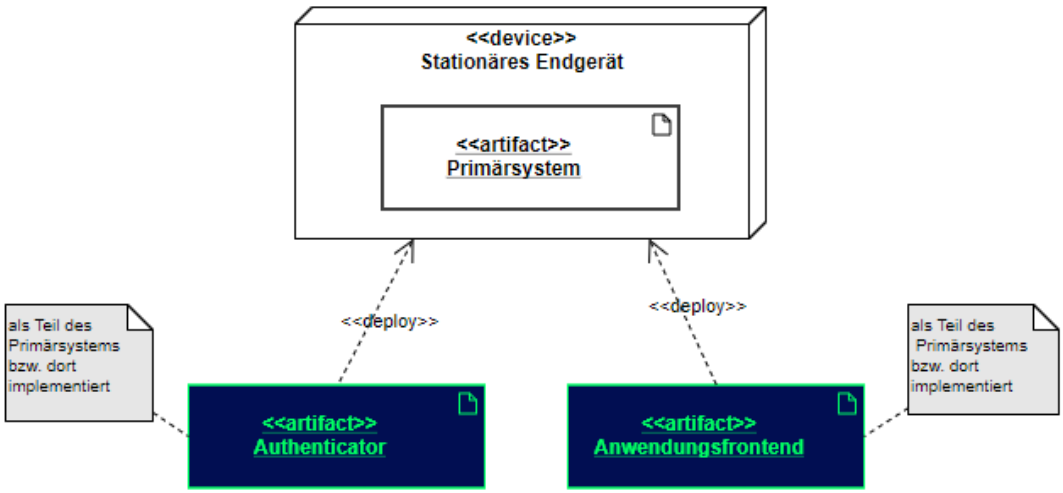

Abbildung 7: Verteilungssicht beim Einsatz eines stationären Endgerätes und eines Primärsystems

In jedem Fall benötigt jeder Nutzer genau ein Endsystem, auf welchem das Authenticator-Modul gemeinsam mit dem Anwendungsfrontend installiert und eingerichtet ist.

9 Anhang A – Interaktionen mit Smartcards der TI

Das folgende Kapitel hat einen rein informativen Charakter und soll Hilfestellung beim Verständnis der Interaktion mit Smartcards der TI geben. Es ergeben sich aus den Beschreibungen keine normativen Anforderungen, welche umzusetzen wären. Normative Vorgaben aus referenzierten Dokumenten bleiben führend. Als Ziel soll hiermit ein grundsätzliches Verständnis der Abläufe vermittelt werden, welches eine Realisierung der Kommunikation des Authenticator-Moduls mit den Smartcards erleichtert.

9.1 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.Soll die Challenge durch die SMC-B signiert werden, wird hierbei die Funktion "externalAuthenticate" des PTV-Konnektors verwendet, wenn sich die SMC-B im freigeschalteten Zustand befindet.

9.2 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. Der zur Identität gehörende private Schlüssel wird zum Signieren einer Challenge genutzt. Die Signatur ist die zugehörige Response.

Aus Sicht einer eGK oder eines HBA stellt sich die User Story wie folgt dar (siehe Abbildung 8):

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 8: Betrachtet wird hier ein System, welches aus folgenden Komponenten besteht:

1. Auf einem mobilen Gerät (in Abbildung 8 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 8 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.

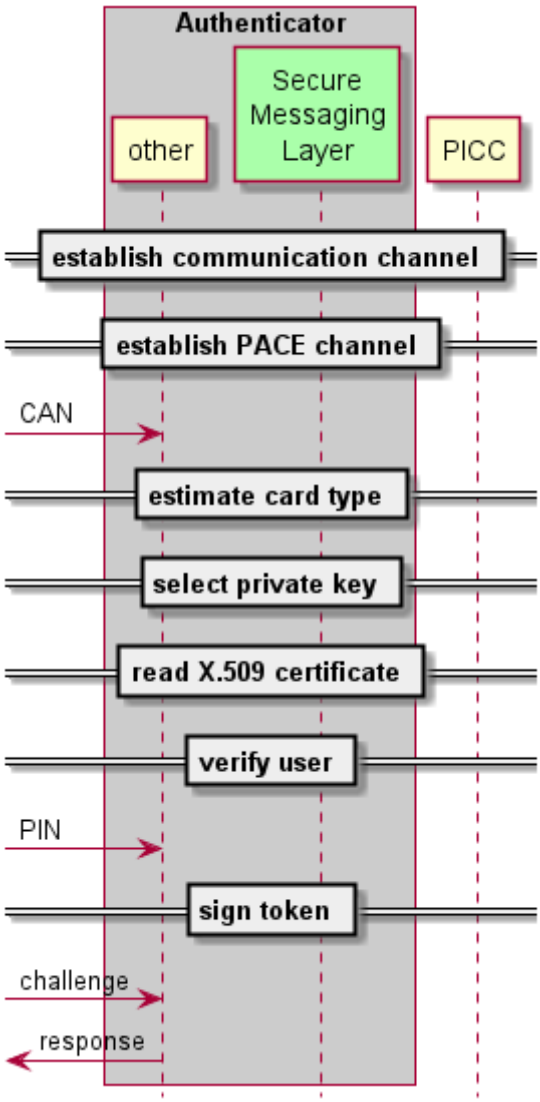

Abbildung 8: Überblick zum Ablauf

9.3 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.

9.3.1 Aufbau eines Kommunikationskanals zwischen Authenticator-Moduls und PICC

Wie ein Kommunikationskanal zwischen Authenticator-Modul und der 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.

9.3.2 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.

9.3.2.1 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.

9.3.2.2 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 9.3.2.1 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.

9.3.2.3 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 2: Zusammenhang CosA_8da mit der folgenden Abbildung

[COS#CosA_8da]	Abbildung 9	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

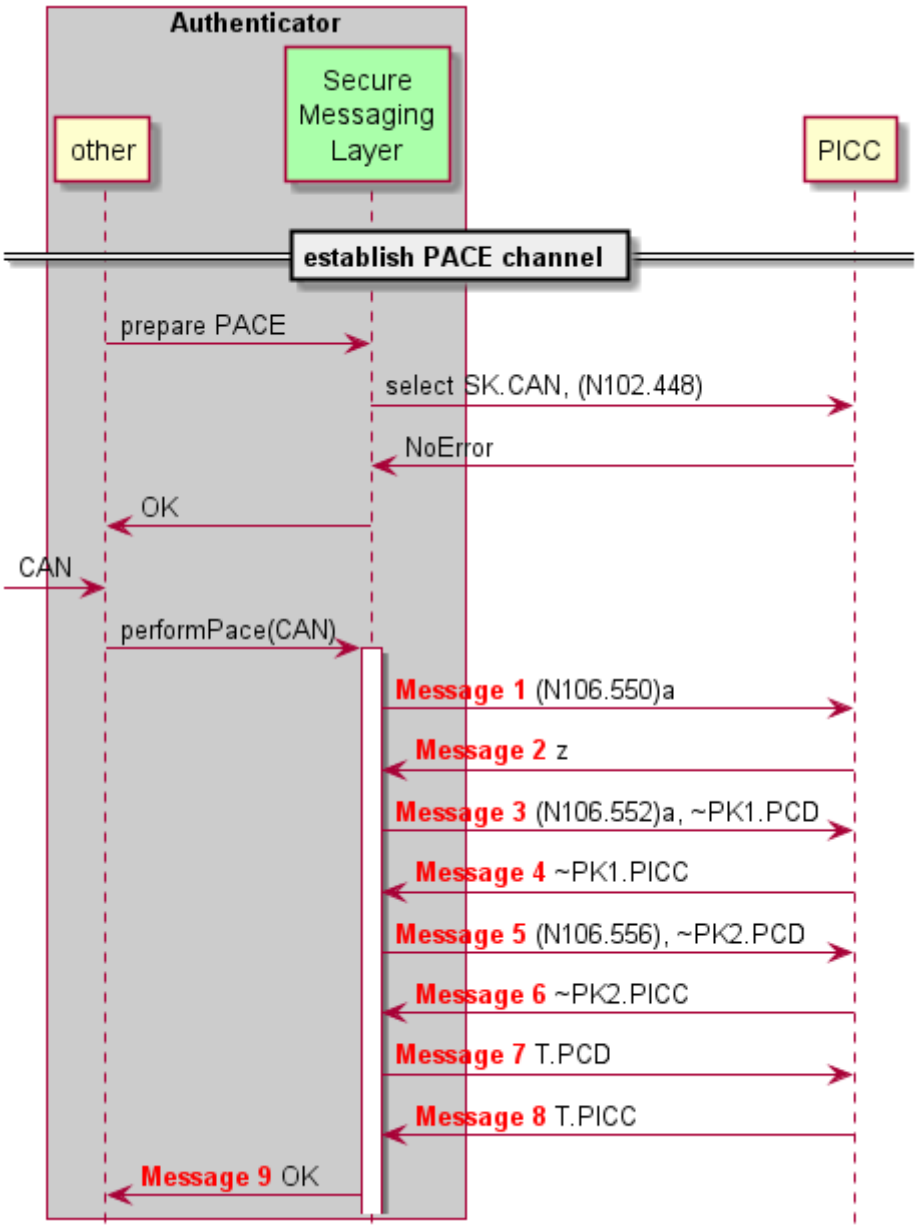

Abbildung 9: Sequenzdiagramm PACE-Authentisierung

Die PICC ist mit einer CAN (Card Access Number) ausgestattet. Die CAN ist auf dem Kartenkörper aufgedruckt. (Für die eGK siehe gemSpec_eGK_Opt - z.B. Abb_eGKOPT_5. Für den HBA gibt es keine Vorgaben der gematik zum Layout, mit Card-G2-A_2038 nur eine Pflicht der Bedruckung). 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 in der 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] zur 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] zur 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)] zur 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)] zur 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.

9.3.3 Ermitteln des Kartentyps

In der vorliegenden Dokumentversion dieser Spezifikation gilt die hier beschriebene Ermittlung des Kartentyps, der in [gemSpec_eGK_ObjSys_G2.1], [gemSpec_HBA_ObjSys] und [gemSpec_HBA_ObjSys_G2.1] definiert ist. Andere Kartentypen der TI (derzeit sind das SMC-B, gSMC-K und gSMC-KT) unterstützen das kontaktlose Interface nicht und werden hier nicht betrachtet.

Unter anderem gibt es folgende Möglichkeiten, den Kartentyp 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 Kartentyp 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.

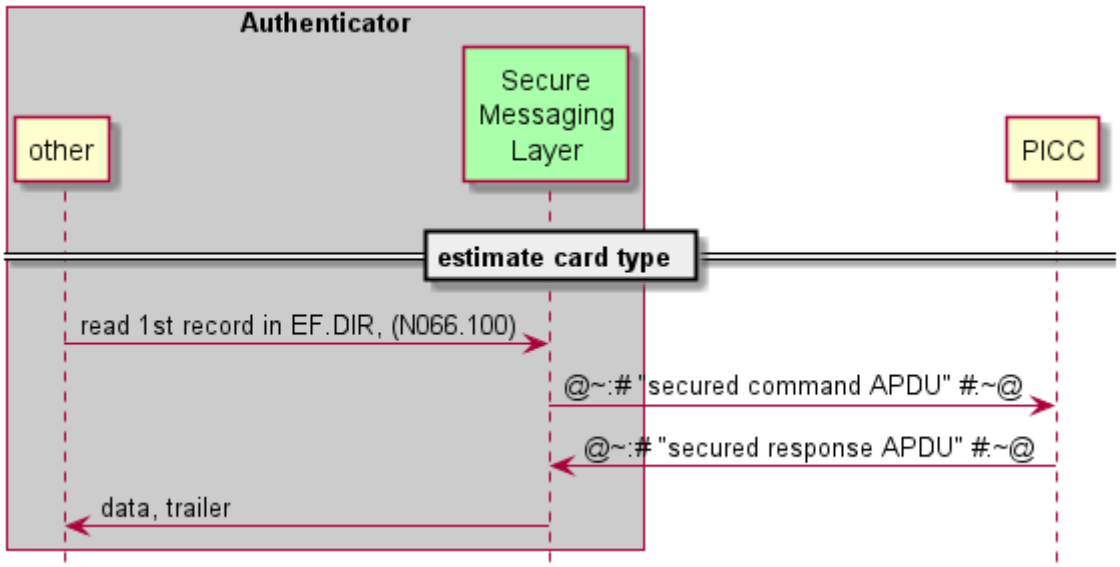

Abbildung 10: Ablauf zur Ermittlung des Kartentyps

Die Kommando-Application Protocol Data Unit (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.

9.3.4 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 9.3.2 und 9.3.3 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. In9.3.3 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 9.3.3 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.

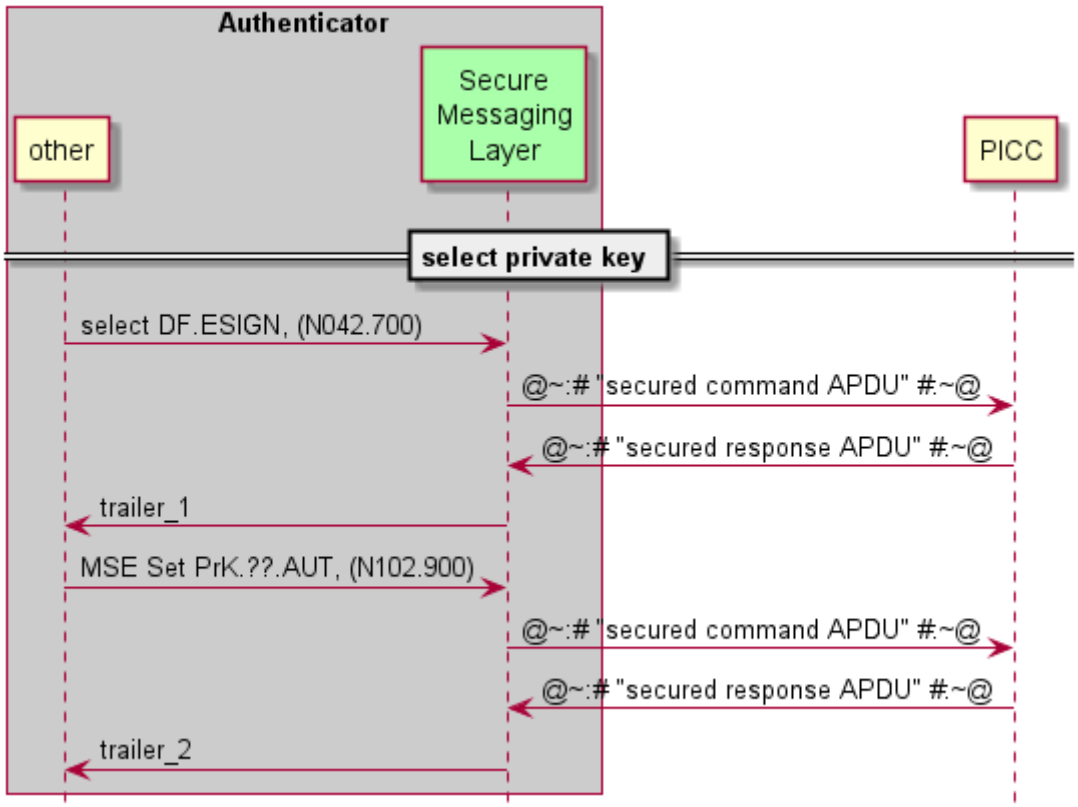

Abbildung 11: Ablauf zur Selektion des privaten Schlüssels

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*.

9.3.5 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 ist. 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: Es 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 erhält, 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.

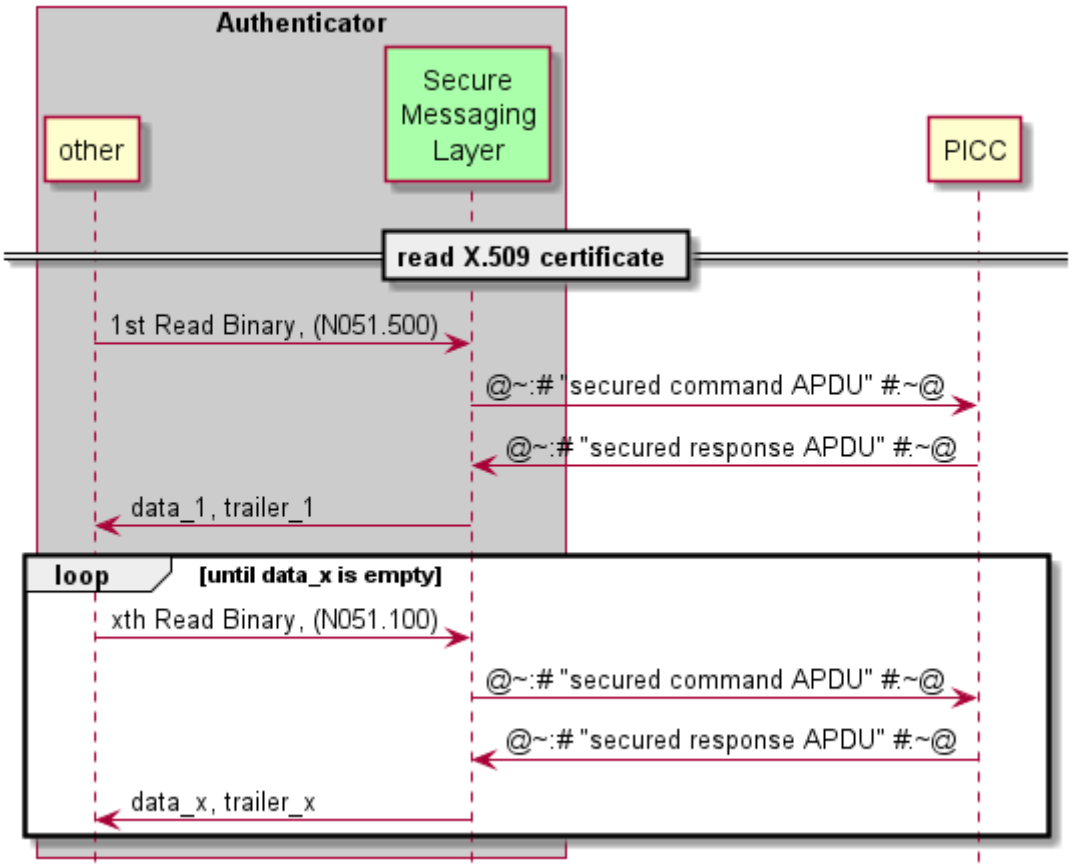

Abbildung 12: Ablauf zum Auslesen des X.509-Zertifikates

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 9.3.3 und des ausgewählten privaten Schlüssels gemäß 9.3.4 zu wählen.

Tabelle 3: Auswahl des shortFileIdentifier

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 Bytes 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 Kartentyp und unabhängig vom ausgewählten privaten Schlüsselobjekt):

Tabelle 4: Kommando APDU für Lesekommandos

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 12 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.

9.3.6 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.

Tabelle 5: Beispielhafte Umwandlungen einer PIN

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

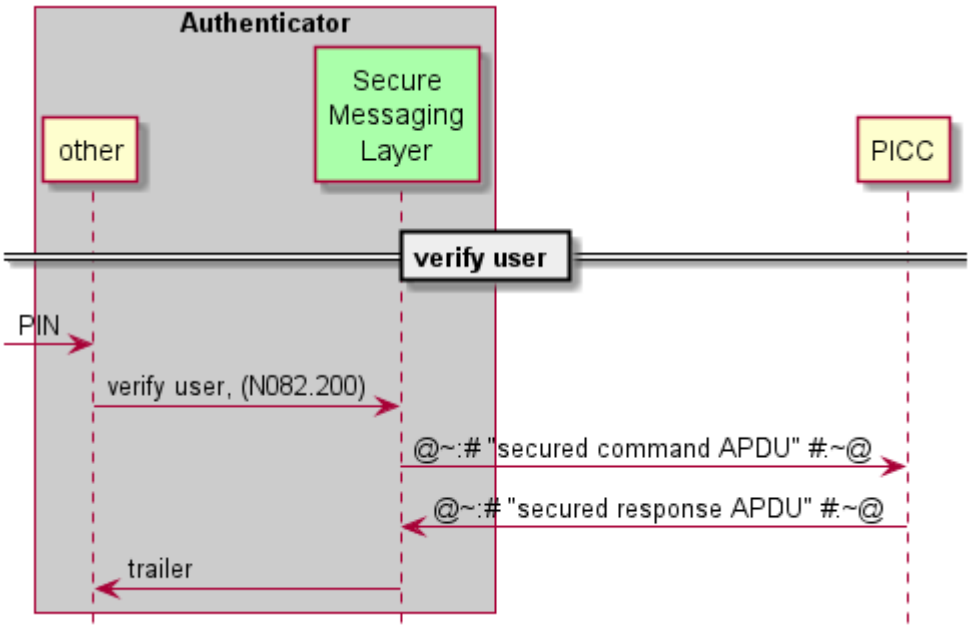

Abbildung 13: Ablauf zur Verifikation des Benutzers

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

Tabelle 6: Passwortobjekt in Abhängigkeit des Kartentyps

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.

9.3.7 Signieren

Typischerweise sind per digitaler Signatur geschützten 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.

9.3.7.1 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 9.3.7.3 weiterverarbeitet wird.

9.3.7.2 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 9.3.7.3 weiterverarbeitet wird.

9.3.7.3 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 9.3.7.1 oder 9.3.7.2 den SHA-256 Hashwert zu *challenge* gemäß *mHash* = SHA-256(*challenge*).

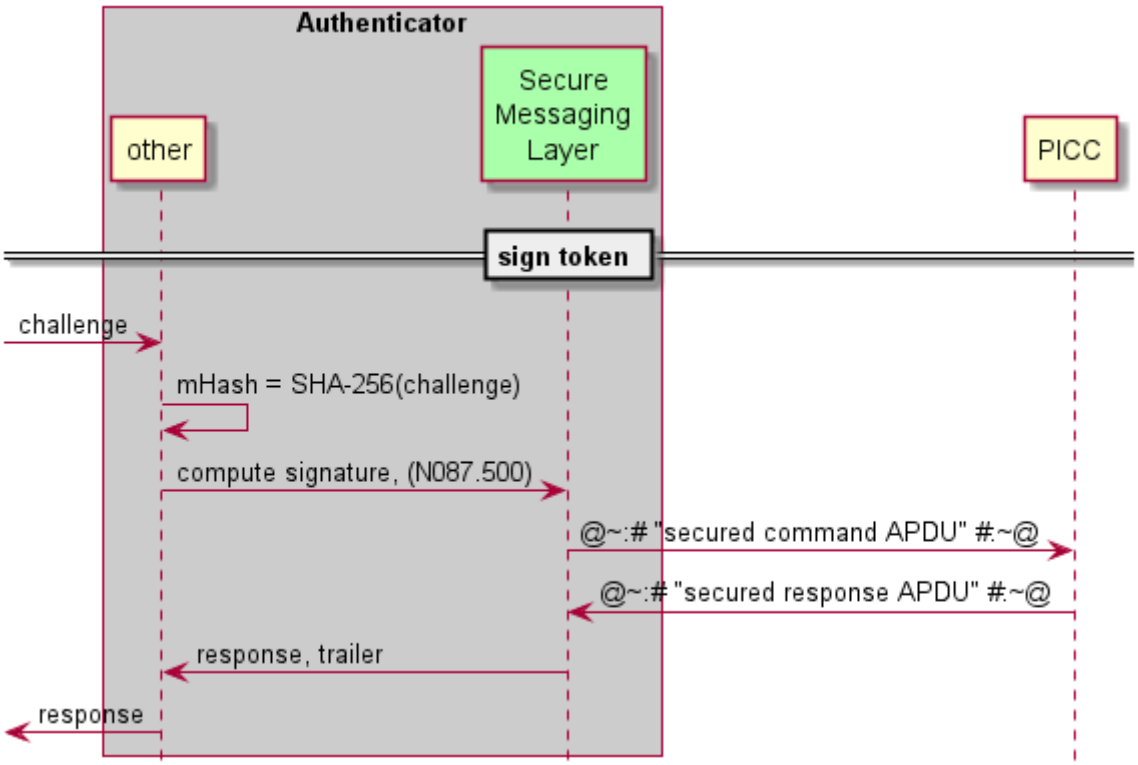

Abbildung 14: Ablauf eines Signaturvorgangs

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.

10 Anhang B – Verzeichnisse

10.1 Abkürzungen

Kürzel	Erläuterung
APDU	Application Protocol Data Unit (Nachricht, die auf der Applikationsebene zwischen einer Smartcard und der externen Welt ausgetauscht wird)
AVS	Apothekenverwaltungssystem
CAN	Card Access Number (Kartenzugriffsnummer, auf dem Kartenkörper aufgedruckte Ziffernfolge)
eGK	Elektronische Gesundheitskarte
FCP	File Control Parameter (Liste mit Eigenschaften einer Datei, die auf einer Smartcard gespeichert ist)
GdV	Gerät des Versicherten
HBA	Heilberufsausweis
IdP	Identity Provider
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWS	JSON Web Signature
JWT	JSON Web Token
KIS	Krankenhausinformationssystem
NFC	Near Field Communication (Kommunikation im Nahfeld einer Antenne)
OAuth 2.0	Open Authorization 2.0
OIDC	OpenID Connect OpenID Connect
PACE	Password Authenticated Connection Establishment (Passwort authentisierter Verbindungsaufbau)
PIN	Personal Identification Number

PICC	Proximity Integrated Circuit Card (kontaktlose Smartcard)
PKI	Public Key Infrastructure
PVS	Praxisverwaltungssystem, ärztliche oder zahnärztliches
QES	Qualifizierte Elektronische Signatur
SMC-B	Security Module Card Typ B, Institutionenkarte
TI	Telematikinfrastruktur
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

10.2 Glossar

Begriff	Erläuterung
Access Token	Ein Access Token (nach [RFC6749 # section-1.4]) wird vom Client (Anwendungsfrontend) benötigt, um auf geschützte Daten eines Resource Servers zuzugreifen. Die Repräsentation kann als JSON Web Token erfolgen.
Authorization-Endpunkt	Der Authorization Endpunkt ist diejenige Schnittstelle, an welcher der Authenticator die zu validierenden Nutzerinformationen SCOPE, CONSENT-Freigabe und Nutzerzertifikat zwecks Prüfung entgegen nimmt und einen "AUTHORIZATION_CODE" heraus gibt.
Authorization Server	OAuth2-Rolle (siehe [RFC6749 # section-1.1]): Der Authorization Server ist Teil des IdP-Dienstes. Der Server authentifiziert den Resource Owner (Nutzer) und stellt Access Tokens für den vom Resource Owner erlaubten Anwendungsbereich (Scope) für einen Resource Server bzw. eine auf einem Resource Server existierende Protected Resource aus.
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.
Claim	Ein Key/Value-Paar im Payload eines JSON Web Token.

Client	OAuth2-Rolle (siehe [RFC6749 # section-1.1]): Eine Anwendung (Relying Party), die auf geschützte Ressourcen des Resource Owners zugreifen möchte, die vom Resource Server bereitgestellt werden. Der Client kann auf einem Server (Webanwendung), Desktop-PC, mobilen Gerät etc. ausgeführt werden.
Discovery Document	Ein OpenID Connect-Metadatendokument (siehe [openid-connect-discovery 1.0]), das den Großteil der Informationen enthält, die für eine App zum Durchführen einer Anmeldung erforderlich sind. Hierzu gehören Informationen wie z.B. die zu verwendenden URLs und der Speicherort der öffentlichen Signaturschlüssel des Dienstes.
Funktionsmerkmal	Der Begriff beschreibt eine Funktion oder auch einzelne, eine logische Einheit bildende Teilfunktionen der TI im Rahmen der funktionalen Zerlegung des Systems.
ID-Token	Ein auf JSON basiertes und nach [RFC7519] (JWT) genormtes Identitäts-Token, mit dem ein Client (Anwendungsfrontend) die Identität eines Nutzers überprüfen kann.
Open Authorization 2.0	Ein Protokoll zur Autorisierung für Web-, Desktop und mobile Anwendungen. Dabei wird es einem Endbenutzer (Resource Owner) ermöglicht, einer Anwendung (Client) den Zugriff auf Daten oder Dienste (Resources) zu ermöglichen, die von einem Dritten (Resource Server) bereitgestellt werden.
OpenID Connect	OpenID Connect (OIDC) ist eine Authentifizierungsschicht, die auf dem Autorisierungsframework OAuth 2.0 basiert. Es ermöglicht Clients, die Identität des Nutzers anhand der Authentifizierung durch einen Autorisierungsserver zu überprüfen (siehe [openid-connect-core 1.0]).
JSON Web Token	Ein auf JSON basiertes und nach [RFC7519] (JWT) genormtes Access Token. Das JWT ermöglicht den Austausch von verifizierbaren Claims innerhalb seines Payloads.
Resource Owner	OAuth2-Rolle (siehe [RFC6749 # section-1.1]): Eine Entität (Nutzer), die einem Dritten den Zugriff auf ihre geschützten Ressourcen gewähren kann. Diese Ressourcen werden durch den Resource Server bereitgestellt. Ist der Resource Owner eine Person, wird diese als Nutzer bezeichnet.
Resource Server	OAuth2-Rolle (siehe [RFC6749 # section-1.1]): Der Server (Dienst), auf dem die geschützten Ressourcen (Protected Resources) liegen. Er ist in der Lage, auf Basis von Access Tokens darauf Zugriff zu gewähren. Ein solcher Token repräsentiert die delegierte Autorisierung des Resource Owners.

SSO-Token	Gegen Vorlage eines gültigen SSO Token ist keine erneute Nutzerauthentifizierung für die Ausstellung eines Access Tokens am IdP-Dienst nötig.
Token-Endpunkt	Der Token-Endpunkt ist die Schnittstelle des IdP-Dienstes an welcher Anwendungsfrontends und Fachdienste gegen Vorlage des "AUTHORIZATION_CODE" ein "ACCESS_TOKEN" und ggf. weitere Token erhält.

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

10.3 Abbildungsverzeichnis

10.4 Tabellenverzeichnis

10.5 Referenzierte Dokumente

10.5.1 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: Glossar der Telematikinfrastruktur
[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]	gematik: Spezifikation Identity Provider-Dienst
[gemSpec_IDP_FD]	gematik: Spezifikation Identity Provider-Frontend
[gemSpec_Krypt]	gematik: Übergreifende Spezifikation: Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur
[gemSpec_OM]	gematik:Übergreifende Spezifikation Operations und Maintenance
[gemSpec_eGK_ObjSys_G2.1]	gematik: Spezifikation der elektronischen Gesundheitskarte, eGK-Objektsystem

10.5.2 Weitere Dokumente

Die weiteren zu beachtenden Dokumente sind folgender Tabelle zu entnehmen.

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories (Juni 2002) ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf

[openid-connect-core]	OpenID Connect Core 1.0 (November 2014) https://openid.net/specs/openid-connect-core-1_0.html
[openid-connect-discovery]	OpenID Connect Discovery 1.0 (November 2014) https://openid.net/specs/openid-connect-discovery-1_0.html
[RFC6749]	The OAuth 2.0 Authorization Framework (Oktober 2012) https://tools.ietf.org/html/rfc6749
[RFC6750]	The OAuth 2.0 Authorization Framework: Bearer Token Usage (Oktober 2012) https://tools.ietf.org/html/rfc6750
[RFC7231]	Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content (Juni 2014) https://tools.ietf.org/html/rfc7231
[RFC7515]	JSON Web Signature (Mai 2015) https://tools.ietf.org/html/rfc7515
[RFC7516]	JSON Web Encryption (Mai 2015) https://tools.ietf.org/html/rfc7516
[RFC7519]	JSON Web Token (Mai 2015) https://tools.ietf.org/html/rfc7519
[RFC7523]	JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants (Mai 2015) https://tools.ietf.org/html/rfc7523
[RFC7636]	Proof Key for Code Exchange by OAuth Public Clients (September 2015) https://tools.ietf.org/html/rfc7636
[RFC8252]	OAuth 2.0 for Native Apps (Oktober 2017) https://tools.ietf.org/html/rfc8252
[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 (Juni 2018) 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 (September

	2018) https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03116/index_htm.html