Elektronische Gesundheitskarte und Telematikinfrastruktur

Feature: Tokenverschlüsselung

Version:	1.0.0 CC
Revision:	356821
Stand:	20.04.2021
Status:	zur Abstimmung freigegeben
Klassifizierung:	öffentlich_Entwurf
Referenzierung:	gemF_Tokenverschlüsselung

Dokumentinformationen

Beim vorliegenden Dokument handelt es sich um einen Entwurf in Vorbereitung auf zukünftige normative Festlegungen als Grundlage entsprechender Zulassungs- und Bestätigungsverfahren. Die gematik versendet diesen Entwurf mit dem Ziel, dass sich Interessierte vorab einen Überblick zur möglichen Weiterentwicklung der Anwendung elektronische Patientenakte verschaffen können.

Die gematik übernimmt keine Gewähr für Aktualität, Richtigkeit und Vollständigkeit dieses Entwurfs. Die gematik behält sich das Recht vor, ohne vorherige Ankündigung Änderungen oder Ergänzungen vorzunehmen oder von den Regelungen insgesamt oder teilweise Abstand zu nehmen.

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
0.0.1			initiale Erstellung	gematik
1.0.0 CC	20.04.21		zur Abstimmung freigegeben	gematik

Inhaltsverzeichnis

1 Einordnung des Dokuments

1.1 Zielgruppe

Das Dokument bildet alle Schritte des Entwicklungsprozesses in verschiedenen Kapiteln ab. Daher unterscheidet sich die intendierte Zielgruppe zwischen den einzelnen Kapiteln.

Das Kapitel 2 betrachtet die fachliche Ebene. Es dient der fachlichen Abstimmung mit Stakeholdern und fachlichen Verbänden.

Kapitel 3 und 4 beschreiben die konkrete Lösung und deren Auswirkung auf Produkttypen. Sie sind daher hauptsächlich für die Abstimmung mit Herstellern, Anbietern und deren Auftraggebern relevant.

1.2 Abgrenzungen

Das Dokument umfasst im Kapitel 3 und 4 nur Änderungen an Spezifikationen/Steckbriefen der gematik und ist daher als Ergänzung zur entsprechenden Spezifikation der gematik zu verstehen und zu lesen. Sollten als Teil eines Features neue Produkttypen eingeführt werden, so wird deren Spezifikation in einem neuen Dokument erfolgen und hier nur referenziert werden. Das neue Dokument für den Produkttyp wird dann ergänzend zur Feature-Spezifikation verteilt.

1.3 Methodik

1.3.1 Epic und User Story

Epics und zugeordnete User Stories werden durch eine eindeutige ID gekennzeichnet.

Epic und UserStory werden im Dokument wie folgt dargestellt:
<Jira-ID> - <Zusammenfassung des Jira-Issue>
Text / Beschreibung
[<=]

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

1.3.2 Anforderungen

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID sowie die dem RFC 2119 [RFC2119] entsprechenden, in Großbuchstaben geschriebenen

deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet.

Da in dem Beispielsatz „Eine leere Liste DARF NICHT ein Element besitzen." die Phrase „DARF NICHT" semantisch irreführend wäre (wenn nicht ein, dann vielleicht zwei?), wird in diesem Dokument stattdessen „Eine leere Liste DARF KEIN Element besitzen." verwendet. Die Schlüsselworte werden außerdem um Pronomen in Großbuchstaben ergänzt, wenn dies den Sprachfluss verbessert oder die Semantik verdeutlicht.

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

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

Nicht alle in diesem Dokument gelisteten Anforderungen sind neu oder verändert. Einige bestehen unverändert weiter aber werden aufgrund ihres Bezugs zur Thematik für das Gesamtverständnis gelistet.

2 Epic und User Story

Für alle Artefakte, welche zwischen Anwendungsfrontend, Authenticator-Modul, IDP-Dienst und Fachdienst übertragen werden und personenbezogene Daten enthalten, soll eine über die TLS-Verbindung hinausgehende Verschlüsselung auf Anwendungsebene vorgesehen werden.

2.1 User Stories

USt-1 - Berücksichtigung der Datenobjekte für die zustandslose Gestaltung des IDP-Dienstes

Da der Smartcard-basierte IDP-Dienst ohne eigenen Datenbestand realisiert wird, müssen die für die einzelnen Operationsschritte notwendigen Daten in den Anfragen transportiert werden. Um hierbei die umfassende Verschlüsselung zu gewährleisten, sind Anpassungen notwendig.

USt-2 - Verzicht auf eine Fachdienst-spezifische Verschlüsselung durch den IDP-Dienst

Die an den E-Rezept-Fachdienst übermittelten ACCESS-Token sollen nicht noch zusätzlich durch den IDP-Dienst verschlüsselt werden. Eine Übertragung durch den VAU-Kanal ist ausreichend. Es besteht kein Grund, vor der Anwendung Informationen zu verbergen, welche ihr durch das ID-Token ohnehin zur Verfügung stehen und für Use Cases in der Anwendung benötigt werden.

Die betrieblichen Risiken eines Schlüsselmanagements zwischen IDP und Fachdienst werden hiermit beseitigt und die Möglichkeit zur späteren Pseudonymisierung des Fachdienstes geschaffen.

3 Spezifikation

Es werden folgende neue und geänderte Anforderungen und ergänzende Texte in die jeweiligen Dokumente aufgenommen.

Einige wenige der genannten Anforderungen sind unverändert geblieben, sie werden aber für eine einheitliche Gesamtsicht zur Thematik ebenfalls in diesem Dokument aufgeführt.

3.1 gemSpec_IDP_Dienst

3.2 Begriffsdefinition

Tabelle: TAB_IDP_DIENST_0003 Bezeichnungen der extern genutzten Schlüssel und Adressen des IDP-Dienstes

	PUK	private Key	URI Dienst
Authorization-Endpunkt (AUTH)	PuK_IDP_SIG - für die Signaturprüfung des „CHALLENGE_TOKEN“,des "AUTHORIZATION_CODE" und des "SSO_TOKEN" - kodiert in einem FD.SIG-Zertifikat PuK_IDP_ENC - für die Verschlüsselung der signierten Challenge durch das Authenticator-Modul	PrK_IDP_SIG - zum Signieren des „CHALLENGE_TOKEN“ des "AUTHORIZATION_CODE" und des "SSO_TOKEN" PrK_IDP_ENC - zum Entschlüsseln der signierten Challenge	URI_AUTH (authorization_endpoint) URI_AUTH_SSO (sso_endpoint)

Token-Endpunkt (TOKEN)	PuK_IDP_SIG - für die Signaturprüfung des "ID_TOKEN" und des "ACCESS_TOKEN" - kodiert in einem FD.SIG-Zertifikat PuK_IDP_ENC - für die Verschlüsselung des "KEY_VERIFIER" durch das Anwendungsfrontend	PrK_IDP_SIG - zum Signieren des "ID_TOKEN" und des "ACCESS_TOKEN" PrK_IDP_ENC - für die Entschlüsselung des "KEY_VERIFIER"	URI_TOKEN (token_endpoint)
Pairing-Endpunkt (PAIR)	PuK_IDP_ENC - für die Verschlüsselung des ACCESS_TOKEN für den Pairing-Vorgang	PrK_IDP_ENC - für die Entschlüsselung des ACCESS_TOKEN für den Pairing-Vorgang	URI_PAIR
Discovery-Endpunkt	PuK_DISC_SIG - für die Signaturprüfung des Discovery Document - kodiert in einem FD.SIG-Zertifikat	PrK_DISC_SIG - zum Signieren des Discovery Document	URI_DISC

Bei allen extern genutzten Schlüsseln handelt es sich um ECC-Schlüsselpaare der Kurve brainpoolP256r1. Für IDP_ENC ist im Gegensatz zu den anderen beiden Schlüsseln keine Bestätigung als Zertifikat vorgesehen. Die maximale Einsatzdauer des Schlüsselpaares liegt analog zum IDP_SIG und DISC_SIG bei maximal 5 Jahren.

3.3 Verfahrensbeschreibung

10. Das Authenticator-Modul überträgt die signierte "CHALLENGE" mit dem Smartcard-Zertifikat, verschlüsselt mittels PuK_IDP_ENC, an den IdP-Dienst (Antwort Schritt 7). Falls ein "SSO_TOKEN" beim Authenticator-Modul existiert, wird dieser Token zusammen mit einer unveränderten "Challenge" zum IdP-Dienst transportiert.

11. Der Authorization-Endpunkt entschlüsselt und validiert die signierte Challenge "SESSION_ID", "CHALLENGE" und "SIGNATUR". Die Signatur wird anhand des im "x5c"-Header mitgelieferten Authentifizierungszertifikats der Smartcard validiert. Falls ein "SSO_TOKEN" angenommen wurde, wird dieses validiert. Entschlüsselt wird das "SSO_TOKEN" vom Authorization-Endpunkt mit seinem Schlüsselmaterial, welches er zur

Verschlüsselung genutzt hat. Die Überprüfung der Signatur des "SSO_TOKEN" führt der Authorization-Endpunkt anhand seines öffentlichen Schlüssels "PUK_AUTH" durch.

14. Der Authorization-Endpunkt überträgt den "AUTHORIZATION_CODE" und den "SSO_TOKEN" an das Authenticator-Modul (Antwort Schritt 3). Falls das Authenticator-Modul ein vorhandenes "SSO_TOKEN" an den Authorization-Endpunkt zur Erlangung eines "AUTHORIZATION_CODE" geschickt hat, wird kein neues "SSO_TOKEN" vom Authorization - Endpunkt erstellt und verschickt. Der "AUTHORIZATION_CODE" und das "SSO_TOKEN" werden vom Authorization-Endpunkt mit seinem privaten Schlüssel "PrK_IDP_SIG" signiert. Der Authorization-Endpunkt verschlüsselt das "SSO_TOKEN" für sich und den "AUTHORIZATION_CODE" für den Token-Endpunkt. Das zur Verschlüsselung verwendete Schlüsselmaterial muss die Vorgaben der [gemSpec_Krypt] beachten.

16. Das Anwendungsfrontend erzeugt sich einen AES256-"Token-Key", verknüpft ihn mit dem "CODE_VERFIER" zum "KEY_VERIFIER" und sendet diesen unter Nutzung des öffentlichen Schlüssels PUK_IDP_ENC verschlüsselt zusammen mit dem "AUTHORIZATION_CODE" zum Token-Endpunkt des IDP-Dienstes.

17. Der Token-Endpunkt entschlüsselt den "AUTHORIZATION_CODE" und validiert ihn anhand des öffentlichen Schlüssels "PUK_IDP_SIG" des Authorization-Endpunktes.

18. Der Token-Endpunkt entschlüsselt und validiert den "KEY_VERIFIER", entnimmt aus diesem den "CODE_VERIFIER" und gleicht diesen mit der "CODE_CHALLENGE" aus dem "AUTHORIZATION_CODE" ab.

19. Der Token-Endpunkt erzeugt die erforderlichen Token, signiert sie mit seinem privaten Schlüssel "PrK_IDP_SIG" und verschlüsseltsie mit dem „Token-Key" des Anwendungsfrontend, welchen er dem „KEY_VERIFIER" entnimmt.

21. Das Anwendungsfrontend entschlüsselt die Token mit seinem „Token-Key" und prüft die Token-Signatur anhand des öffentlichen Schlüssels "PUK_IDP_SIG" des Token-Endpunktes.

22. Das Anwendungsfrontend reicht das gültige "ACCESS_TOKEN" auf Anwendungsebene verschlüsselt beim Fachdienst ein.

23. Der Fachdienst entschlüsselt das "ACCESS_TOKEN" entsprechend dem für diese Anwendung vorgesehenen Verfahren.

3.12 Aufgabe des Fachdienstes

Der Fachdienst nimmt das zwischen Frontend und Fachdienst anwendungsspezifisch verschlüsselte "ACCESS_TOKEN" entgegen.

4 Zerlegung des Produkttyps

A_20687-01 - Bereitstellung der öffentlichen Schlüsselteile
Der Authorization Server MUSS zu allen verwendeten privaten Schlüsseln "PrK_IDP_SIG", "PrK_IDP_ENC" und "PrK_DISC_SIG" das öffentliche Pendant "PuK_IDP_SIG", "PuK_IDP_ENC" und "PuK_DISC_SIG" zum Download bereitstellen. Dies ermöglicht die Prüfung der von den einzelnen Schnittstellen vorgenommenen Signaturen ebenso wie die zielgerichtete Verschlüsselung des Payload für den bestimmten Empfänger.**[<=]**

5.1.1 Aufbau des Discovery Document

A_20458-02 - Inhalte des Discovery Document
Der Discovery-Endpunkt MUSS sowohl im internen, als auch im externen Discovery Document gemäß [RFC8414 # section-2] mindestens die folgenden Attribute als URI angeben:

- "issuer" (hier ist der IdP-Dienst erreichbar)
- "jwks_uri" (für den Abruf von „PUK_IDP_ENC" sowie des öffentlichen Schlüssels und des Zertifikats von „PUK_IDP_SIG" entsprechend TAB_IDP_DIENST_0003 [RFC7517] – identifiziert anhand der „kid"-Parameter (puk_idp_enc / puk_idp_sig)
- "uri_disc" (URI, unter welcher das Discovery Document bereitgestellt wird)
- "authorization_endpoint" (URI des Dienstes und des öffentlichen Verschlüsselungsschlüssels des Authorization-Endpunktes gemäß [RFC6749])
- "sso_endpoint" (URI des Authorization-Endpunktes für Requests mit SSO-Token)
- "auth_pair_endpoint" (URI des Authorization-Endpunktes für Requests mit Pairing-Daten)
- "token_endpoint" (URI des Token-Endpunktes gemäß [RFC6749])
- "uri_puk_idp_enc" und „uri_puk_idp_sig" (URI der JWK Objekte für die zwei Schlüssel und des Zertifikates).

[<=]

Der genaue Aufbau entspricht gemSpec_IDP-Dienst#Kapitel 7.7

5.1.2 Erneuerung des Discovery Document

A_20691-01 - Das Discovery Document ist maximal 24 Stunden alt
Der Discovery-Endpunkt MUSS das Discovery Document regelmäßig alle 24 Stunden oder nach durchgeführten Änderungen umgehend neu erstellen, mit dem `"PrK_DISC_SIG"` signieren und am mit der gematik vereinbarten Downloadpunkt `"URI_DISC"` bereitstellen.**[<=]**

A_20592 entfällt

5.1.3 Schutz des Discovery Document

A_20591-01 - Festlegungen zur Signatur der Discovery Documents
Der IdP-Dienst MUSS die Signatur der Discovery Documents dabei durch die Verwendung einer JSON Web Signature (JWS) [RFC7515 # section-3 - Compact Serialization] und des Schüssels PrK_DISC_SIG gewährleisten. Als Algorithmus ist dementsprechend "BP256R1" zu wählen.
Der IdP-Dienst MUSS bei der Signaturerstellung das Signaturzertifikat des PUK_DISC_SIG im x5c Claim einbetten.**[<=]**

5.2.1 Authorization Server Eingangsdaten

A_20699-03 - Annahme von CHALLENGE_TOKEN: Authentication_Data-Struktur
Der Authorization-Endpunkt MUSS

- entweder das mit dem Zertifikat der Smartcard des Nutzers signierte und durch das Authenticator-Modul mittels PuK_IDP_ENC verschlüsselte "CHALLENGE_TOKEN"
- oder – im Fall der Verwendung von alternativen Authentisierungsmitteln – die mit dem privaten Schlüssel des Endgeräts signierte und durch das Authenticator-Modul mittels PuK_IDP_ENC verschlüsselte Authentication_Data-Struktur

annehmen. Er MUSS in beiden Fällen anhand des im Header vorhandenen "exp"-Claim die Token anhand der Systemzeit auf zeitliche Gültigkeit prüfen. Sind diese nicht mehr gültig, MUSS der Authentifizierungsvorgang abgebrochen werden. Im Fall der Gültigkeit MÜSSEN diese mit dem PrK.IDP.ENC entschlüsselt werden.**[<=]**

Der Aufbau der Anfrage entspricht [gemSpec_IDP-Dienst#Kapitel 7.3].

Hinweis: Als Verschlüsselungsalgorithmus ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

A_20951-01 - Validierung der Signatur und des Zertifikats des CHALLENGE_TOKEN
Der Authorization-Endpunkt MUSS die Signatur des vom Authenticator-Modul übertragenen, signierten `CHALLENGE_TOKEN` anhand des mitgelieferten Authentifizierungs-Zertifikats überprüfen. Die Überprüfung MUSS neben der Signatur auch das Authentifizierungszertifikat anhand von OCSP umfassen. **[<=]**

Der genaue Aufbau des vom Authenticator-Modul übertragenen, signierten `CHALLENGE_TOKEN` findet sich in [gemSpec_IDP-Dienst#Kapitel 7.3].

A_20948-01 - Validierung des "SSO_TOKEN"
Der Authorization-Endpunkt MUSS den angenommenen und entschlüsselten "`SSO_TOKEN`" validieren. Die Validierung MUSS die Überprüfung der Signatur anhand seines öffentlichen Schlüssels PuK_IDP_SIG und die Überprüfung der zeitlichen Gültigkeit des "SSO_TOKEN" anhand des Attributs "`auth_time`" umfassen.**[<=]**

5.2.2 Authorization-Endpunkt Ausgangsdaten

A_21317 - Verschlüsselung des " AUTHORIZATION_CODE"
Der Authorization-Endpunkt des IDP-Dienst MUSS den "AUTHORIZATION_CODE " für den Token-Endpunkt mit eigenem Schlüsselmaterial verschlüsseln, welches den Anforderungen aus gemSpec_Krypt genügt.**[<=]**

A_21330 - Ablaufzeitpunkt von "AUTHORIZATION_CODE" und "SSO_TOKEN"
Der Authorization-Endpunkt des IDP-Dienst MUSS im exp Claim des jeweiligen JWE Headers den Ablaufzeitpunkt des "AUTHORIZATION_CODE" bzw. "SSO_TOKEN" liefern.**[<=]**

Der Aufbau der Header entspricht [gemSpec_IDP-Dienst#Kapitel 7.4].

A_20521-02 - Inhalt des CHALLENGE_TOKEN an das Authenticator-Modul
Der IdP-Dienst MUSS die ihm vorliegenden Session-Informationen (z.B. "`SESSION_ID`", "`CODE_CHALLENGE`", "`SCOPE`" und alle Informationen über Anwendungsfrontend und Authenticator-Modul) mit seinem privaten Schlüssel "`PrK_IDP_SIG`" und der technischen Rolle „oid_idpd" gemäß [gemSpec_OID # Abschnitt 3.5.4] signieren und als JWT ergänzt um die "`USER_CONSENT`"-Anfrage an das Authenticator-Modul senden. Als Algorithmus ist dementsprechend "BP256R1" zu wählen.**[<=]**

5.3.1 Token-Endpunkt Eingangsdaten

A_20321-01 - Annahme und Prüfung von "AUTHORIZATION_CODE" und "KEY_VERIFIER"
Der Token-Endpunkt MUSS den vom Anwendungsfrontend übertragenen "AUTHORIZATION_CODE", und den "KEY_VERIFIER" des Anwendungsfrontend annehmen. Der „AUTHORIZATION_CODE" ist dabei mittels eines durch den IDP-Dienst für Authorization-Endpunkt und Token-Endpunkt definierten Verfahren zu entschlüsseln. **[<=]**

Der Aufbau der Anfrage entspricht [gemSpec_IDP-Dienst#Kapitel 7.5].

Hinweis: Als Verschlüsselungsalgorithmus für den "Key_VERIFIER" ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

A_21318 - Prüfung des "AUTHORIZATION_CODE"
Der Token-Endpunkt MUSS die Signatur des "AUTHORIZATION_CODE" unter Verwendung des Schlüssels PuK_IDP_SIG prüfen.**[<=]**

A_21319 - Prüfung des CODE_VERIFIER
Der Token-Endpunkt MUSS den "CODE_VERIFIER" aus dem mittels PuK_IDP_ENC verschlüsselten "KEY_VERIFIER" extrahieren und die Überprüfung gegen die "CODE_CHALLENGE" mit S256 (Algorithmus nach [RFC7636 # section-4.2]) durchführen.**[<=]**

Der Aufbau des "KEY_VERIFIER" entspricht [gemSpec_IDP-Dienst#Kapitel 7.5].

A_21320 - Entschlüsseln des „Token-Key"
Der Token-Endpunkt MUSS den „Token-Key" aus dem mittels PuK_IDP_ENC verschlüsselten "KEY_VERIFIER" extrahieren.**[<=]**

Der Aufbau des "KEY_VERIFIER" entspricht [gemSpec_IDP-Dienst#Kapitel 7.5].

5.3.2 Token-Endpunkt Ausgangsdaten

A_20695-01 - Signieren des "SSO_TOKEN"
Der Authorization-Endpunkt MUSS den "SSO_TOKEN" mit seinem eigenen privaten Schlüssel "PrK_IDP_SIG" signieren. Als Algorithmus ist dementsprechend "BP256R1" zu wählen.**[<=]**

A_20327-02 - Signatur des "ID_TOKEN" und "ACCESS_TOKEN"
Der Token-Endpunkt MUSS alle erstellten "ID_TOKEN" und "ACCESS_TOKEN", um deren Integrität sicherzustellen und eine eineindeutige Erklärung über deren Herkunft abzugeben, mit seinem privaten Schlüssel "PrK_IDP_SIG" signieren. [RFC7523 # section-3 Spiegelpunkt 9] ist zu gewährleisten. Als Algorithmus ist dementsprechend "BP256R1" zu wählen.**[<=]**

Zum Aufbau des Signatur-Header siehe [gemSpec_IDP-Dienst#Kapitel 7.6].

A_21321 - Verschlüsselung von "ACCESS_TOKEN" und „ID_TOKEN"
Der Token-Endpunkt MUSS "ACCESS_TOKEN" und „ID_TOKEN", nach der Signatur, mittels JWE [RFC7516]) unter Nutzung des „Token-Key" des Anwendungsfrontend verschlüsseln.**[<=]**

Zum Aufbau des Verschlüsselungs-Header siehe [gemSpec_IDP-Dienst#Kapitel 7.6].

A_20328 entfällt

7 Anhang B

Folgendes Kapitel verdeutlicht beispielhaft den Aufbau der einzelnen Objekte und Aufrufe, welche im Rahmen der Beantragung von ACCESS-TOKEN und ID-TOKEN beim IDP-Dienst generiert und ausgetauscht werden.

Die einzelnen Aufrufe werden den Schritten aus Kapitel 3.3 zugewiesen und in entsprechenden Unterkapiteln dargestellt.

Diese Beispiele ersetzen die vorher an verschiedenen Stellen des Dokumentes verteilten Darstellungen.

Die gematik wird weitere Beispielsätze auf ihrer Webseite in jeweils aktueller Form bereitstellen

7.1 Authorization Request

2. Das Anwendungsfrontend überträgt die "`CODE_CHALLENGE`" gemäß [RFC8252 # Anhang B] an das Authenticator-Modul.

3. Das Authenticator-Modul überträgt die "`CODE_CHALLENGE`" mit der genutzten "`code_challenge_method`" S256 weiter an den Authorization-Endpunkt des IdP-Dienstes.

/sign_response?
scope=openid+e-rezept
Der Scope entspricht dem zwischen E-Rezept-Fachdienst und IDP festgelegten Wert. Mit diesem antwortet der E-Rezept-Fachdienst bei fehlendem ACCESS_TOKEN und http-Statuscode 401.
&response_type=code
Referenziert den erwarteten Response-Type des Flows. Muss immer 'code' lauten. Damit wird angezeigt, dass es sich hierbei um einen Authorization Code Flow handelt. Für eine nähere Erläuterung siehe OpenID-Spezifikation.
&redirect_uri=http%3A%2F%2Fredirect.gematik.de%2Ferezept
Die URL wird vom Primärsystem beim Registrierungsprozess im IDP hinterlegt und leitet die Antwort des Servers an diese Adresse um.
&state=AcYxMQ5MZMpRh6WOBjs8
Dieser Parameter wird vom Client zufällig generiert, um CSRF zu verhindern. Indem der Server mit diesem Wert antwortet, werden Redirects legitimiert.
&code_challenge_method=S256
Das Primärsystem generiert einen Code-Verifier und erzeugt darüber einen Hash im Verfahren SHA-256, hier abgekürzt als S256. Teil von PKCE.
&nonce=nN4LkW1moAwg1tofYZtf
String zur Verhinderung von CSRF-Attacken. Dieser Wert ist optional. Wenn er mitgegeben wird, muss der gleiche Wert im abschließend ausgegebenen ID-Token wieder auftauchen.
&client_id=eRezeptApp
Die Client-ID des Primärsystems wird beim Registrieren des Primärsystems beim IDP festgelegt.
&code_challenge=SU8xsVcUypYGUi2g-mzs7rvR2lMtQ9vyj_9Hxs0WcII
Der Hashwert des Code-Verifier wird zum IDP als Code-Challenge gesendet. Teil von PKCE.

7.2 Authorization Response

7. Der Authorization-Endpunkt überträgt "CHALLENGE" und Consent-Abfrage "USER_CONSENT" zum Authenticator-Modul.

200
Cache-Control=no-store,
Pragma=no-cache,

Version=0.1-SNAPSHOT,
Content-Type=application/json,
Transfer-Encoding=chunked,
Date=<Zeitpunkt der Antwort. Beispiel Fri, 05 Feb 2021 12:46:20 GMT>
Keep-Alive=timeout=60,
Connection=keep-alive

Response-Body:

```
{
  "challenge": "eyJhbGciOiJC...",
  "user_consent": {
    "requested_scopes": {
      "e-rezept": "Zugriff auf die E-Rezept-Funktionalität.",
      "openid": "Zugriff auf den ID-Token."
    },
    "requested_claims": {
      "organizationName": "Zustimmung zur Verarbeitung der
Organisationszugehörigkeit",
      "professionOID": "Zustimmung zur Verarbeitung der Rolle",
      "idNummer": "Zustimmung zur Verarbeitung der ID (z.B.
Krankenversichertennummer, Telematik-ID)",
      "given_name": "Zustimmung zur Verarbeitung des Vornamens",
      "family_name": "Zustimmung zur Verarbeitung des Nachnamens"
    }
  }
}
```

Challenge Token:

```
{
  "alg": "BP256R1",
  "typ": "JWT",
  "kid": "puk_idp_sig"
}
```

```
{
  "iss": "http://url.des.idp",
  "response_type": "code",
  "snc": "[server-nonce. Wird verwendet, um noise hinzuzufügen. Beispiel:
\"yvqCVQO09TFsFfaJ312RfYoi1oII0BHtIDIRpzD8Gu0\"]",
  "code_challenge_method": "S256",
  "token_type": "challenge",
  "nonce": "nN4LkW1moAwg1tofYZtf",
  "client_id": "eRezeptApp",
  "scope": "openid e-rezept",
```

```
  "state": "AcYxMQ5MZMpRh6WOBjs8",
  "redirect_uri": "http://redirect.gematik.de/erezept",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244172]",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243992]",
  "code_challenge": "SU8xsVcUypYGUi2g-mzs7rvR2lMtQ9vyj_9Hxs0WcII",
  "jti": "[A unique identifier for the token, which can be used to prevent reuse of the
token. Value is a case-sensitive string. Beispiel: \"07925bdc7c5d9990\"]"
}
```

7.3 Authentication Request

10. Das Authenticator-Modul überträgt "CHALLENGE" mit dem Smartcard-Zertifikat an den IdP-Dienst (Antwort Schritt 7).

https://<FQDN Server>/<AUTHORIZATION_ENDPOINT>

Multiparts:
signed_challenge=<signierter und anschließend verschlüsselter Challenge Token>

Challenge Response (Encryption Header):

```
{
  "alg": "ECDH-ES",
  "enc": "A256GCM",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244172]",
  "cty": "NJWT",
  "epk": {
    "kty": "EC",
    "x": "LgkJSQwrz1bCoFjSLhay9O7TLaQImYW7jeOF6XmpQX4",
    "y": "dTC6ri-f1QqpJp7M4LLg0lw4FzrzNc29nrrzjPwEWWc",
    "crv": "BP-256"
  }
}
```

```
{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Die verschlüsselte Challenge Response."
}
```

Challenge Response (Decrypted):

```
{
  "typ": "JWT",
  "cty": "NJWT",
  "alg": "BP256R1",
  "x5c": [
    "[Enthält das verwendete Signer-Zertifikat als Base64 ASN.1 DER-Encoding. Hier
kommt ausnahmsweise NICHT URL-safes Base64-Encoding zum Einsatz!]"
  ]
}
```

```
{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Dieses Token ist die vom Server in der
vorigen Nachricht übergebene Challenge. Sie muss exakt wie empfangen auch wieder
```

übertragen werden. "
}

7.3.1 Authentication Request (SSO)

https://<FQDN Server>/<SSO_ENDPOINT>

Multiparts:
sso_token=<SSO-Token des IDP>

unsigned_challenge = < unveränderter Challenge Token>

7.4 Authentication Response

Der Authorization-Endpunkt überträgt den "AUTHORIZATION_CODE" und den "SSO_TOKEN" an das Authenticator-Modul (Antwort Schritt 3).

302
Cache-Control=no-store,
Pragma=no-cache,
Location=http://redirect.gematik.de/erezept?
code=<Authorization Code in Base64-URL-Safe Encoding. Wird unten detaillierter aufgeführt>
Der Authorization-Code. Er berechtigt zur Abholung eines Access Token. Er ist vom IDP für den IDP verschlüsselt und dementsprechend vom Client nicht weiter zu verarbeiten.
&state=AcYxMQ5MZMpRh6WOBjs8
Der state der Session. Sollte dem zufällig generierten state-Wert aus der initialen Anfrage entsprechen.
&ssotoken=<SSO-Token in Base64-URL-Safe Encoding. Wird unten detaillierter aufgeführt>
Der SSO-Token. Mit diesem kann der Client sich wiederholt einloggen, ohne erneut den Besitz der Karte durch Unterschreiben einer Challenge beweisen zu müssen. Er ist vom IDP für den IDP verschlüsselt und dementsprechend vom Client nicht weiter zu verarbeiten. - Nicht im Fall der Anmeldung über ein Primärsystem.
Content-Length=0,
Date=<Zeitpunkt der Antwort. Beispiel Fri, 05 Feb 2021 12:46:20 GMT>
Keep-Alive=timeout=60,
Connection=keep-alive

Response-Body:

Die Inhalte von Authorization Code und SSO Token sind IDP-spezifisch und nicht normativ vorgegeben. Sie dienen hier nur dem Verständnis.

Authorization Code (Encryption Header):
{
  "alg": "dir",
  "enc": "A256GCM",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244053]",

```
  "cty": "NJWT"
}

{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Der Authorization Code"
}


Authorization Code (Decrypted):
{
  "alg": "BP256R1",
  "typ": "JWT",
  "kid": "puk_idp_sig"
}
{
  "organizationName": "AOK Plus",
  "professionOID": "1.2.276.0.76.4.49",
  "idNummer": "X114428530",
  "iss": "http://url.des.idp",
  "response_type": "code",
  "snc": "[server-nonce. Wird verwendet, um noise hinzuzufügen. Beispiel:
\"Ay6WUqtAUcV2p9WZYHPo\"]",
  "code_challenge_method": "S256",
  "given_name": "Juna",
  "token_type": "code",
  "nonce": "nN4LkW1moAwg1tofYZtf",
  "client_id": "eRezeptApp",
  "scope": "openid e-rezept",
  "auth_time": "[Timestamp der Authentisierung. Beispiel: 1618243993]",
  "redirect_uri": "http://redirect.gematik.de/erezept",
  "state": "AcYxMQ5MZMpRh6WOBjs8",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244053]",
  "family_name": "Fuchs",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243993]",
  "code_challenge": "SU8xsVcUypYGUi2g-mzs7rvR2lMtQ9vyj_9Hxs0WcII",
  "jti": "[A unique identifier for the token, which can be used to prevent reuse of the
token. Value is a case-sensitive string. Beispiel: \"6e8a61e316472f3b\"]"
}


SSO Token (Encryption Header):
{
  "alg": "dir",
  "enc": "A256GCM",
  "exp": "[Gültigkeit des Token. Beispiel: 1618287193]",
  "cty": "NJWT"
}

{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Das SSO Token"
}


```

SSO Token (Decrypted):

```
{
  "alg": "BP256R1",
  "typ": "JWT",
  "kid": "puk_idp_sig"
}
{
  "organizationName": "AOK Plus",
  "professionOID": "1.2.276.0.76.4.49",
  "auth_time": "[Timestamp der Authentisierung. Beispiel: 1618243993]",
  "idNummer": "X114428530",
  "iss": null,
  "cnf": {
    "x5c": [
      "[Enthält das verwendete Signer-Zertifikat als Base64 ASN.1 DER-Encoding. Hier kommt ausnahmsweise NICHT URL-safes Base64-Encoding zum Einsatz!]"
    ],
    "kid": "844508318621525",
    "kty": "EC",
    "crv": "BP-256",
    "x": "dTXa6yPKCjIr9MbVFxeaLEu82xSCsRrfwcIrLpFqBCs",
    "y": "AJGsJ1cCyGEpCH0ss8JvD4OAHJS8IMm1_rM59jliS-1O"
  },
  "given_name": "Juna",
  "exp": "[Gültigkeit des Token. Beispiel: 1618287193]",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243993]",
  "family_name": "Fuchs"
}
```

7.4.1 Authentication Response (SSO Flow)

Der Authorization-Endpunkt überträgt den "AUTHORIZATION_CODE" an das Authenticator-Modul (Antwort Schritt 3).

```
302
Cache-Control=no-store,
Pragma=no-cache,
Location=https://<FQDN Server>/<TOKEN_ENDPOINT>
   ?code=<Authorization Code des IDP>
   &state=<OAuth 2.0 state value. Constant over complete flow. Value is a case-sensitive string. Beispiel: 'mlE7xX1ysbZQ4Of5YiZ8'>,
Content-Length=0,
Date=<Zeitpunkt der Antwort. Beispiel Fri, 05 Feb 2021 12:46:20 GMT>
Keep-Alive=timeout=60,
Connection=keep-alive
```

7.5 Token Request

16. Das Anwendungsfrontend sendet "CODE_VERFIER" und "AUTHORIZATION_CODE" zum Token-Endpunkt des IDP-Dienstes.

https://<FQDN Server>/<TOKEN_ENDPOINT>
Multiparts:
client_id=eRezeptApp
&code=<Authorization Code des IDP>
&grant_type=authorization_code
&key_verifier=<verschlüsselter KEY_VERIFIER>
&redirect_uri=http%3A%2F%2Fredirect.gematik.de%2Ferezept

Key_verifier (Encryption Header):

```
{
  "alg": "ECDH-ES",
  "enc": "A256GCM",
  "cty": "JSON",
  "epk": {
    "kty": "EC",
    "x": "jqrGQlZqCGQgK7OtJj0gfWPWSbStracf_PreBrA05Lc",
    "y": "V4bbOGUgr-AV6NFLnPJNYkyPLcR_1QkGlR6w4bK1wdI",
    "crv": "BP-256"
  }
}
```

Key_verifier (Body)

```
{
  "token_key": "T0hHOHNKOTFaREcxTmN0dVRKSURraTZxNEpheGxaUEs",
  "code_verifier": "W91A37hQ8oeDRVpnkYgpYthjl4LqYy95A87ISy9zpUM"
}
```

7.6 Token Response

20. Der Token-Endpunkt überträgt die Token an das Anwendungsfrontend (Antwort Schritt 16).

200
Cache-Control=no-store,
Pragma=no-cache,
Version=0.1-SNAPSHOT,
Content-Type=application/json,
Transfer-Encoding=chunked,
Date=<Zeitpunkt der Antwort. Beispiel Fri, 05 Feb 2021 12:46:20 GMT>
Keep-Alive=timeout=60,
Connection=keep-alive

Response-Body:

```
{
  "expires_in": 300,
```

```
  "token_type": "Bearer",
  "id_token": <Mit dem Token_Key verschlüsseltes ID_TOKEN.>
  "access_token": <Mit dem Token_Key verschlüsseltes ACCESS_TOKEN.>
}

Access Token (Encryption Header):
{
  "alg": "dir",
  "enc": "A256GCM",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244294]",
  "cty": "NJWT"
}
{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Das Access Token"
}


Access Token (Decrypted):
{
  "alg": "BP256R1",
  "typ": "at+JWT",
  "kid": "puk_idp_sig"
}
{
  "sub": "[subject. Base64(sha256(audClaim + idNummerClaim + serverSubjectSalt)).
Beispiel: \"ez4D403gBzH1IhnYOXA4aUU-7spqPbWUyUELPoA79CM\"]",
  "professionOID": "1.2.276.0.76.4.49",
  "organizationName": "AOK Plus",
  "idNummer": "X114428530",
  "amr": [
    "mfa",
    "sc",
    "pin"
  ],
  "iss": null,
  "given_name": "Juna",
  "client_id": "eRezeptApp",
  "acr": "gematik-ehealth-loa-high",
  "aud": "https://erp.telematik.de/login",
  "azp": "eRezeptApp",
  "scope": "openid e-rezept",
  "auth_time": "[Timestamp der Authentisierung. Beispiel: 1618243993]",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244294]",
  "family_name": "Fuchs",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243994]",
  "jti": "[A unique identifier for the token, which can be used to prevent reuse of the
token. Value is a case-sensitive string. Beispiel: \"c0bf3cebe428e3c9\"]"
}


ID Token (Encryption Header):
```

```
{
  "alg": "dir",
  "enc": "A256GCM",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244294]",
  "cty": "NJWT"
}
{
  "njwt": "[Ein verschachtelt enthaltenes JWT] - Das ID Token"
}


ID Token (Decrypted):
{
  "alg": "BP256R1",
  "typ": "JWT",
  "kid": "puk_idp_sig"
}
{
  "at_hash": "[Erste 16 Bytes des Hash des Authentication Token
Base64(subarray(Sha256(authentication_token), 0, 16)). Beispiel: \"5AZmDxrYImUa6-
kjMNAL3g\"]",
  "sub": "[subject. Base64(sha256(audClaim + idNummerClaim + serverSubjectSalt)).
Beispiel: \"ez4D403gBzH1IhnYOXA4aUU-7spqPbWUyUELPoA79CM\"]",
  "organizationName": "AOK Plus",
  "professionOID": "1.2.276.0.76.4.49",
  "idNummer": "X114428530",
  "amr": [
    "mfa",
    "sc",
    "pin"
  ],
  "iss": null,
  "given_name": "Juna",
  "nonce": "nN4LkW1moAwg1tofYZtf",
  "aud": "eRezeptApp",
  "acr": "gematik-ehealth-loa-high",
  "azp": "eRezeptApp",
  "auth_time": "[Timestamp der Authentisierung. Beispiel: 1618243993]",
  "scope": "openid e-rezept",
  "exp": "[Gültigkeit des Token. Beispiel: 1618244294]",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243994]",
  "family_name": "Fuchs",
  "jti": "[A unique identifier for the token, which can be used to prevent reuse of the
token. Value is a case-sensitive string. Beispiel: \"c1c760ca67fe1306\"]"
}
```

7.7 Aufbau des Discovery Document

```
{
  "alg": "BP256R1",
  "kid": "puk_disc_sig",
  "x5c": [
    "[Enthält das verwendete Signer-Zertifikat als Base64 ASN.1 DER-Encoding. Hier
```

```
kommt ausnahmsweise NICHT URL-safes Base64-Encoding zum Einsatz!]"
  ]
}
{
  "authorization_endpoint": "[URL des Authorization Endpunkts.]",
  "auth_pair_endpoint": "[URL des Biometrie-Authorization-Endpunkts.]",
  "sso_endpoint": "[URL des SSO-Authorization Endpunkts.]",
  "uri_pair": "[URL des Pairing-Endpunkts.]",
  "token_endpoint": "[URL des Authorization-Endpunkts.]",
  "uri_disc": "[URL des Discovery Document.]",
  "issuer": null,
  "jwks_uri": "[URL einer JWKS-Struktur mit allen vom Server verwendeten Schlüsseln]",
  "exp": "[Gültigkeit des Token. Beispiel: 1618330390]",
  "iat": "[Zeitpunkt der Ausstellung des Token. Beispiel: 1618243990]",
  "uri_puk_idp_enc": "http://url.des.idp/idpEnc/jwk.json",
  "uri_puk_idp_sig": "http://url.des.idp/idpSig/jwk.json",
  "subject_types_supported": [
    "pairwise"
  ],
  "id_token_signing_alg_values_supported": [
    "BP256R1"
  ],
  "response_types_supported": [
    "code"
  ],
  "scopes_supported": [
    "openid",
    "e-rezept"
  ],
  "response_modes_supported": [
    "query"
  ],
  "grant_types_supported": [
    "authorization_code"
  ],
  "acr_values_supported": [
    "gematik-ehealth-loa-high"
  ],
  "token_endpoint_auth_methods_supported": [
    "none"
  ],
  "code_challenge_methods_supported": [
    "S256"
  ]
}
```

3.2 gemSpec_IDP_FD

Die bisher an verschiedenen Stellen des Dokumentes verteilten Beispiele werden ersetzt durch Referenzen auf die entsprechenden Stellen in [gemSpec_IDP-Dienst#Anhang B].

4 Registrierung des Fachdienstes beim IdP-Dienst

A_20296 entfällt

6 "ACCESS_TOKEN"

A_20363-01 - "ACCESS_TOKEN" sind verschlüsselt
Der Fachdienst MUSS die für ihn vom Anwendungsfrontend verschlüsselten "ACCESS_TOKEN" entsprechend dem für diese Übertragung vorgesehenen Verfahren entschlüsseln.
[<=]

Hinweis: Hierbei können je nach Anwendung unterschiedliche Verfahren zum Einsatz kommen. Im Fall des E-Rezeptes wird der "ACCESS_TOKEN" im Rahmen des VAU-Protokolls übertragen und ist damit ausreichend geschützt.

A_20364 - Unverschlüsselt eingehende ACCESS_TOKEN sind ungültig
Fachdienste DÜRFEN unverschlüsselt eingehende `"ACCESS_TOKEN"` NICHT annehmen.
[<=]

A_20365-01 - Die Signatur des "ACCESS_TOKEN" ist zu prüfen
Fachdienste MÜSSEN die Signatur der `"ACCESS_TOKEN"` gegen den öffentlichen Schlüssel des Token-Endpunktes `"PUK_IDP_SIG"` prüfen. Fachdienste MÜSSEN den öffentlichen Schlüssel „PUK_IDP_SIG" dabei dem Discovery Document des IdP-Dienstes entnehmen.**[<=]**

A_21521 - Fachdienst: Prüfung der Signatur des Discovery Document
Fachdienste MÜSSEN 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.**[<=]**

3.3 gemSpec_IDP_Frontend

Die bisher an verschiedenen Stellen des Dokumentes verteilten Beispiele werden ersetzt durch Referenzen auf die entsprechenden Stellen in gemSpec_IDP-Dienst#Anhang B.

6.1 Vorbereitende Maßnahmen

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.**[<=]**

6.2.1 Schnittstellendefinition

A_20700-06 - Authenticator-Modul: Signatur der "CHALLENGE"
Das Authenticator-Modul MUSS die vom Authorization-Endpunkt empfangene CHALLENGE_TOKEN mit dem Zertifikat C.CH.AUT aus der Smartcard des Nutzers signieren. Hierbei wird der über die `CHALLENGE_TOKEN` gebildete HASH-Wert zur Signatur überreicht (siehe Abschnitt 9.3.7.3 Signiervorgang in diesem Dokument).**[<=]**

A_20526-01 - Authenticator-Modul: Response auf das CHALLENGE_TOKEN des Authorization-Endpunktes
Das Authenticator-Modul MUSS das eingereichte "CHALLENGE_TOKEN" mittels JWS (JSON Web Signature) mit der Smartcard signieren, und das Authentifizierungszertifikat der verwendeten Smartcard als x5c Parameter einbetten, dieses Objekt mittels JWE (JSON Web Encryption) mit dem öffentlichen Schlüssel des Authorization-Endpunktes "PUK_IDP_ENC" verschlüsseln und in Form eines HTTP-POST-Requests an den Authorization-Endpunktes senden.

Der Aufbau der der Anfrage und des der einzureichenden Objekte entspricht gemSpec_IDP-Dienst#Kapitel 7.3**[<=]**

Hinweis: Das Signieren und Verschlüsseln des "`CHALLENGE_TOKEN`" ist durch die Verwendung eines Nested JWT[angelehnt an den folgenden Draft: https://tools.ietf.org/html/draft-yusef-oauth-nested-jwt-03] zu realisieren. Im cty-Header ist "NJWT" zu setzen, um anzuzeigen, dass es sich um einen Nested JWT handelt. Das Signieren wird dabei durch die Verwendung einer JSON Web Signature (JWS) [RFC7515 # section-Compact Serialization] gewährleistet. Die Verschlüsselung des signierten Token wird durch die Nutzung der JSON Web Encryption (JWE) [RFC7516 # section-3] sichergestellt. Als Verschlüsselungsalgorithmus ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

A_21322 - Authenticator-Modul - sichere Speicherung des "SSO-TOKEN"
Das Authenticator-Modul MUSS empfangene SSO-Token gegen unberechtigten Zugriff schützen.**[<=]**

7.1 Vorbereitende Maßnahmen

A_20501 entfällt

7.2.1 Schnittstellendefinition

A_20624 entfällt

A_21323 - Erzeugung des " Token-Key"
Das Anwendungsfrontend MUSS vor dem Abrufen von ID_Token und ACCESS_Token einen zufälligen 256bit AES Schlüssel ("Token-Key") erzeugen. **[<=]**

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_21324 - Erzeugen des „KEY_VERIFIER"
Das Anwendungsfrontend MUSS dem "KEY_VERIFIER" bilden indem "Token-Key" und "CODE_VERIFIER" in einem JSON Objekt kodiert werden.

Der Aufbau des "KEY_VERIFIER" entspricht gemSpec_IDP-Dienst#Kapitel 7.5**[**<=**]**

A_20529-01 - Senden von "AUTHORIZATION_CODE" und "KEY_VERIFIER" an den Token-Endpunkt
Das Anwendungsfrontend MUSS den "KEY_VERIFIER" mittels JWE (JSON Web Encryption (JWE) [RFC7516 # section-3]) und PUK_IDP_ENC verschlüsseln und zusammen mit dem "AUTHORIZATION_CODE" TLS-gesichert als HTTP/1.1 POST Request an den Token-Endpunkt senden.

Die Aufbau der Anfrage entspricht gemSpec_IDP-Dienst#Kapitel 7.5**[**<=**]**

Hinweis: Als Verschlüsselungsalgorithmus für den "KEY_VERIFIER" ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

A_19938-01 - Annahme des ID_TOKEN
Das Anwendungsfrontend MUSS das vom Token-Endpunkt ausgegebene "ID_TOKEN" als HTTP/1.1 Statusmeldung 200 verarbeiten und mittels " Token-Key" entschlüsseln.
Das Anwendungsfrontend MUSS das "ID_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird oder nicht mit dem vorher übermittelten "Token-Key" verschlüsselt war.

Die Aufbau der Antwort und des ID_TOKEN entspricht gemSpec_IDP-Dienst#Kapitel 7.6**[**<=**]**

A_20283-01 - Annahme des "ACCESS_TOKEN"
Das Anwendungsfrontend MUSS das vom Token-Endpunkt ausgegebene "ACCESS_TOKEN" in der HTTP/1.1 Statusmeldung 200 verarbeiten und mittels " Token-Key" entschlüsseln.
Das Anwendungsfrontend MUSS das "ACCESS_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird oder nicht mit dem vorher übermittelten "Token-Key" verschlüsselt war.

Die Aufbau der Antwort und des ACCESS_TOKEN entspricht gemSpec_IDP-Dienst#Kapitel 7.6**[**<=**]**

A_21325 - Verschlüsselte Übertragung zum Fachdienst
Das Anwendungsfrontend MUSS den entschlüsselten "ACCESS_TOKEN", zusätzlich zur Übertragung mittels TLS, auf Anwendungsebene verschlüsselt zum Fachdienst übertragen. **[**<=**]**

A_21326 - Löschung von ACCESS-TOKEN
Das Anwendungsfrontend MUSS "ACCESS-TOKEN" beim Beenden sowie nach Ablauf ihrer Gültigkeit sicher löschen .**[**<=**]**

A_21327 - Löschung von ID-TOKEN
Das Anwendungsfrontend MUSS „ID-TOKEN" beim Beenden sowie nach Ablauf ihrer Gültigkeit sicher löschen .**[**<=**]**

A_21328 - Sichere Speicherung der Token
Das Anwendungsfrontend MUSS empfangene „ID-TOKEN" und „ACCESS-TOKEN" gegen unberechtigten Zugriff schützen.**[**<=**]**

3.4 gemSpec_FD_eRp

A_19987 entfällt

3.5 gemILF_PS_eRp

5.1.4.1

Die gematik wird Beispielsätze und weitere Hilfestellungen auf ihrer Webseite in jeweils aktueller Form bereitstellen

A_20656-01 - Prüfung der Signatur des Discovery Document
Das Primärsystem MUSS die JWS (JSON Web Signature) [RFC7515 # section-3] - Compact Serialization] Signatur des Discovery Document auf mathematische Korrektheit sowie über die Funktion "VerifyCertificate" des Konnektors gemäß [gemSpec_Kon#4.1.9.5.3] bzw. [gemILF_PS#4.4.4.3] auf Gültigkeit des ausstellenden Zertifikates innerhalb der TI prüfen.

Der genaue Aufbau entspricht gemSpec_IDP-Dienst#Kapitel 7.7**[<=]**

~~Als SignatureType ist urn:ietf:rfc:5652 für eine CMS-Signatur zu verwenden. Weitere optionale Parameter kommen nicht zur Anwendung.~~

A_21337 - Löschung von TOKEN bei zeitlichem Ablauf
Das Primärsystem MUSS vorhandene "ACCESS_TOKEN", "ID_TOKEN" und "AUTHORIZATION_CODE"-Objekte nach Ablauf ihrer Gültigkeit sicher löschen. **[<=]**

A_21338 - Sichere Speicherung der Token
Das Primärsystem MUSS empfangene "ACCESS_TOKEN", "ID_TOKEN" und "AUTHORIZATION_CODE"-Objekte gegen unberechtigten Zugriff schützen. **[<=]**

5.1.4.2

A_20659 - Erzeugen des CODE_VERIFIER
Das Primärsystem MUSS zur Laufzeit einen "CODE_VERIFIER" (Zufallswert) gemäß [RFC7636 # section-4.1] bilden. Der "CODE_VERIFIER" MUSS eine Länge von mindestens 43 und maximal 128 Zeichen enthalten. Dabei sind die folgenden Zeichen zulässig: [A-Z] / [a-z] / [0-9] / "-" / "." / "_" / "~".**[<=]**

A_20660 - Erzeugen des Hash-Werts des CODE_VERIFIER
Das Primärsystem MUSS über den "CODE_VERIFIER" einen SHA256-HASH-Wert, die sogenannte "CODE_CHALLENGE", gemäß [RFC7636 # section-4.2] bilden.
code_challenge = BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))**[<=]**

A_21333 - Erzeugung des "Token-Key"
Das Primärsystem MUSS vor dem Abrufen von ID_Token und ACCESS_Token einen zufälligen 256bit-AES-Schlüssel ("Token-Key") erzeugen.**[<=]**

A_21334 - Erzeugung des "KEY_VERIFIER"
Das Primärsystem MUSS dem "KEY_VERIFIER" bilden, indem " Token-Key" und "CODE_VERIFIER" in einem JSON-Objekt kodiert werden.
Der Aufbau des "KEY_VERIFIER" entspricht gemSpec_IDP-Dienst#Kapitel 7.5.**[<=]**

Der Authorization-Endpunkt legt nun eine "session_id" an, stellt alle nötigen Informationen zusammen und erzeugt das "CHALLENGE_TOKEN".

A_20663-01 - Prüfung der Signatur des CHALLENGE_TOKEN
Das Primärsystem MUSS die Signatur des "CHALLENGE_TOKEN" gegen den aktuellen öffentlichen Schlüssel des Authorization-Endpunktes "PUK_IDP_SIG" prüfen. Liegt dem Primärsystem der öffentliche Schlüssel des Authorization-Endpunktes noch nicht vor, MUSS es diesen gemäß dem „kid"-Parameter "puk_idp_sig" aus dem Discovery Document abrufen.**[<=]**

Das Primärsystem verwendet nun die AUT-Identität der SM-B der LEI und deren Konnektor, um das gehashte "CHALLENGE_TOKEN" des IdP-Dienstes zu signieren.

A_20665-01 - Signatur der Challenge des IdP-Dienstes
Das Primärsystem MUSS für das Signieren des CHALLENGE_TOKEN des IdP-Dienstes mit der Identität ID.HCI.AUT der SM-B die Operation ExternalAuthenticate des Konnektors gemäß [gemSpec_Kon#4.1.13.4] bzw. [gemILF_PS#4.4.6.1] verwenden und als zu signierende Daten `BinaryString` den SHA-256-Hashwert des CHALLENGE_TOKEN in Base64-Codierung übergeben.

Der Aufbau der der Anfrage und des der einzureichenden Objekte entspricht gemSpec_IDP-Dienst#Kapitel 7.3.**[<=]**

A_20666-01 - Auslesen des Authentisierungszertifikates
Das Primärsystem MUSS das Zertifikat ID.HCI.AUT der SM-B über die Operation ReadCardCertificate des Konnektors gemäß [gemSpec_Kon#4.1.9.5.2] bzw. [gemILF_PS#4.4.4.2] auslesen.**[<=]**

A_20667-01 - Response auf die Challenge des Authorization-Endpunktes
Das Primärsystem MUSS das eingereichte "CHALLENGE_TOKEN" zusammen mit der von der Smartcard signierten Challenge-Signatur "signed_challenge" (siehe A_20665) und dem Authentifizierungszertifikat der Smartcard (siehe A_20666), mit dem öffentlichen Schlüssel des Authorization-Endpunktes "PUK_IDP_ENC" verschlüsselt, an diesen in Form eines HTTP-POST-Requests senden.

Der Aufbau der der Anfrage und des der einzureichenden Objekte entspricht gemSpec_IDP-Dienst#Kapitel 7.3**[<=]**

Hinweis: Das Signieren und Verschlüsseln des "CHALLENGE_TOKEN" ist durch die Verwendung eines Nested JWT [angelehnt an den folgenden Draft: https://tools.ietf.org/html/draft-yusef-oauth-nested-jwt-03] zu realisieren. Im cty-Header ist "NJWT" zu setzen, um anzuzeigen, dass es sich um einen Nested JWT handelt. Das Signieren wird dabei durch die Verwendung einer JSON Web Signature (JWS) [RFC7515 # section-3 - Compact Serialization] gewährleistet. Die Verschlüsselung des signierten Token wird durch die Nutzung der JSON Web Encryption (JWE) [RFC7516 # section-3] sichergestellt. Als Verschlüsselungsalgorithmus ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

Der Authorization-Endpunkt validiert nun die "session" sowie die "signed_challenge" und prüft das Zertifikat der LEI. Anschließend verknüpft er die "session" mit der Identität aus dem Authentisierungszertifikat und erstellt einen "AUTHORIZATION_CODE", welchen er als Antwort zurücksendet

Das Primärsystem empfängt nun diesen "AUTHORIZATION_CODE" von IDP-Dienst und reicht ihn zusammen mit dem KEY_VERIFIER beim Token-Endpunkt ein.

A_20669 entfällt

~~Zur Prüfung von Zertifikatstyp-OID und Rollen-OID siehe Hinweis zu A_20657.~~

A_20670 entfällt

~~Anschließend werden der zu Beginn des Prozesses erzeugte "CODE_VERFIER" und der "AUTHORIZATION_CODE" zum Token-Endpunkt des IDP-Dienstes gesendet, um dort gegen "ID_TOKEN" und "ACCESS_TOKEN" eingetauscht zu werden~~

A_20671-01 - Einreichen des AUTHORIZATION_CODE beim Token-Endpunkt
Das Primärsystem MUSS den "Key_Verifier" mittels JWE und PUK_IDP_ENC verschlüsseln und zusammen mit dem "AUTHORIZATION_CODE" TLS-gesichert und als HTTP/1.1 POST Request an den Token-Endpunkt senden.

Die Aufbau der Anfrage entspricht gemSpec_IDP-Dienst#Kapitel 7.5**[<=]**

Als Verschlüsselungsalgorithmus ist ECDH-ES (Elliptic Curve Diffie-Hellman Ephemeral Static key agreement) vorgesehen.

Der Token-Endpunkt validiert den "CODE_VERFIER" und gleicht diesen mit der "code_challenge" ab. Dann erzeugt er die erforderlichen Token und verschlüsselt beide mit dem "Token-Key".

Das Primärsystem erhält nun den signierten "ID_TOKEN"und den ~~für es nicht lesbaren~~ "ACCESS_TOKEN" vom Token-Endpunkt und prüft die Signatur des "ID_TOKEN".

A_20672-01 - Annahme des ID_TOKEN
Das Primärsystem MUSS das vom Token-Endpunkt ausgegebene "ID_TOKEN" als HTTP/1.1 Statusmeldung 200 verarbeiten und mittels " Token-Key" entschlüsseln. Das Primärsystem MUSS das "ID_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird oder nicht mit dem vorher übermittelten " Token-Key" verschlüsselt war.

Die Aufbau der Antwort und des "ID_TOKEN" entspricht gemSpec_IDP-Dienst#Kapitel 7.6**[<=]**

A_20673-01 - Annahme des "ACCESS_TOKEN"
Das Primärsystem MUSS das vom Token-Endpunkt ausgegebene "ACCESS_TOKEN" in der HTTP/1.1 Statusmeldung 200 verarbeiten und mittels "Token-Key" entschlüsseln. Das Primärsystem MUSS das "ACCESS_TOKEN" ablehnen, wenn dieses außerhalb der mit dem Token-Endpunkt etablierten TLS-Verbindung übertragen wird oder nicht mit dem vorher übermittelten " Token-Key" verschlüsselt war.

Die Aufbau der Antwort und des ACCESS_TOKEN entspricht gemSpec_IDP-Dienst#Kapitel 7.6**[<=]**

4 Übersicht Produkt- und Anbietertypen

Die Produkttypsteckbriefe werden wie folgt angepasst.

Für neue Anforderungen werden die nachstehenden Prüfverfahren definiert.

IDP-D

A_20687-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20458-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20591-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20699-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20951-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20948-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21317	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21330	funkt. Eignung: Test Produkt/FA
A_20321-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21318	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21319	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21320	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20695-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20327-02	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21321	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20521-02	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20691-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20328	entfällt

A_20592	entfällt

Anb_IDP-D

A_20592	entfällt

Anb_eRp_FD

A_20296	entfällt
A_19987	entfällt

eRp_FD

A_20363-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20364	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA

eRp_FdV

A_20624	entfällt
A_21323	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21324	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20529-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_19938-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20283-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21325	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21326	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21327	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21328	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA

eRp_FdV_AdV

A_20624	entfällt
A_21323	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA

A_21324	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20529-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_19938-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_20283-01	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21325	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21326	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21327	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA
A_21328	Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA

gemSST_PS_eRp_abgebend und gemSST_PS_eRp_verordnend

A_20669	entfällt
A_20670	entfällt
A_20659	funkt. Eignung: Konformitätsbestätigung
A_20656-01	funkt. Eignung: Konformitätsbestätigung
A_21333	funkt. Eignung: Konformitätsbestätigung
A_21334	funkt. Eignung: Konformitätsbestätigung
A_20665-01	funkt. Eignung: Konformitätsbestätigung
A_20666-01	funkt. Eignung: Konformitätsbestätigung
A_20667-01	funkt. Eignung: Konformitätsbestätigung
A_20671-01	funkt. Eignung: Konformitätsbestätigung
A_20672-01	funkt. Eignung: Konformitätsbestätigung
A_20673-01	funkt. Eignung: Konformitätsbestätigung
A_21337	funkt. Eignung: Herstellererklärung
A_21338	funkt. Eignung: Herstellererklärung