Beim vorliegenden Dokument handelt es sich um einen Entwurf der gematik in Vorbereitung auf zukünftige normative Festlegungen als Grundlage entsprechender Zulassungs- und Bestätigungsverfahren. Die gematik veröffentlicht diesen Entwurf mit dem Ziel, dass sich Interessierte bereits frühzeitig einen Überblick über die mögliche Weiterentwicklung der Telematikinfrastruktur verschaffen können. Die gematik übernimmt keine Gewähr für die Aktualität, Richtigkeit und Vollständigkeit dieses Entwurfes und behält sich das Recht vor, ohne vorherige Ankündigung Änderungen oder Ergänzungen vorzunehmen oder von den Regelungen insgesamt bzw. teilweise Abstand zu nehmen.

Änderungsbedarf in gemSpec_Krypt

MTOM/XOP-Unterstützung bei der Verbindung ePA-FdV/FM ePA <-> VAU (Dokumentenmangement)

Kontext: Erfahrungswert aus der gematik-AKTOR-ePA-Implementierung

Nachdem die Etablierung des VAU-Kanals abgeschlossen ist, kommuniziert ein VAU-Client (bspw. ePA-FdV) mit der VAU bspw., um einen großen verschlüsselten Arztbrief der Akte hinzuzufügen. Dabei ist es für die Performanz (also für die Nutzerakzeptanz) günstig, größere Daten mittels der Kodierung gemäß "SOAP Message Transmission Optimization Mechanism (MTOM)“/„XML-binary Optimized Packaging (XOP)“ zu transportieren. MTOM/XOP ist die nach W3C empfohlene Methode, um über HTTP/SOAP größere binäre Daten zu transportieren. Dabei werden die Binärdaten nicht Base64-kodiert innerhalb einer XML-Datenstruktur kodiert, sondern innerhalb von HTTP über eine MIME-Multipart-Kodierung. Damit wird die Übertragungsrate der Daten rund 27,5% schneller (effizientere Kodierung).

Um den Einsatz von MTOM/XOP zu ermöglichen, wird eine verschlüsselte Übertragung des originär intendierten Content-Type ermöglicht.

6.8 Nutzerdatentransport

A_16945 - VAU-Protokoll: Client, verschlüsselte Kommunikation (1)
Wie bei der Schlüsselaushandlung MUSS der Client mittels HTTP-POST-Request die nun verschlüsselte Kommunikation initiieren. ~~Sei "Plaintext" die zu übermittelnde Nachricht.~~ Der Client MUSS einen unsigned 64-Bit-Nachrichtenzähler führen, den er bei jeder abgeschickten Nachricht um zwei erhöhen MUSS.
Er bildet die Datenstruktur "P1" mit

```
P1=Version (ein Byte mit dem Wert 0x01) ||
   Nachrichtenzähler (unsigned 64-Bit im Big-Endian-Format) ||
   Anzahl der Bytes der folgenden optionalen zusätzlichen HTTP-
Header-Informationen (unsigned 32-Bit im Big-Endian-Format) ||
   optionale HTTP-Header-Informationen ||
   Plaintext
```

wobei „Plaintext“ die zu übertragende Nutzlast (bspw. SOAP-Request) bezeichne.

Wenn die Anzahl der Bytes der folgenden optionalen zusätzlichen HTTP-Header-Informationen mit 0 (also 0x00000000) angegeben wird, so gibt es keine folgenden optionalen zusätzlichen HTTP-Header-Informationen, d. h. es folgen direkt die Plaintext-Bytes.

Verständnishinweis: bez. der optionalen zusätzlichen HTTP-Header-Information vgl. [gemSpec_Krypt#6.11 VAU-Kanal und MTOM/XOP].

Der Nachrichtenzähler MUSS initial mit 1 starten. Der Client MUSS P1 mittels AES-GCM unter Verwendung einer zufällig erzeugten 96-Bit-Nonce verschlüsseln. Der Client berechnet so den "Ciphertext".
Die vom Client nun an den Server zu übermittelnde Datenstruktur MUSS folgende Form besitzen:

```
 256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag
```

Diese Nachricht MUSS der Client per HTTP-POST-Request mit Content-Type 'application/octet-stream' ohne weitere Kodierungen versenden.
[<=]

A_16952 - VAU-Protokoll: Server, verschlüsselte Kommunikation
Der Server erkennt aus der KeyID, welchen AES-Schlüssel er verwenden muss. Falls ihm die KeyID unbekannt ist, so MUSS er mit einer VAUServerError-Nachricht mit der Fehlermeldung "KeyID XXX not found" antworten, wobei er XXX durch die empfangene KeyID in Hexadezimalform ersetzen MUSS.
Falls bei der Entschlüsselung ein Fehler auftritt (bspw. Authentication-Tag passt nicht zur Nachricht), MUSS der Server mit einer VAUServerError-Nachricht mit der Fehlermeldung "AES-GCM decryption error." antworten.

Falls die Entschlüsselung erfolgreich war, MUSS der Server die ersten 64-Bit (8 Byte) als Nachrichtenzähler (unsigned, im Big-Endian-Format) – im Folgenden als Zählerwert bezeichnet -- interpretieren und diese vom folgenden Plaintext entfernen. Der Zählerwert MUSS größer als der letzte auf diese Art empfangene Zählerwert (für die aktuelle KeyID) plus 1 sein. (Zähler + 1 war der Zählerwert der Server-Response.)
Anderenfalls MUSS er

1. die KeyID in seiner Datenbasis verwerfen und alle damit verbundenen Schlüssel sicher löschen,
2. die restlichen Plaintext-Daten verwerfen, und
3. mit einer VAUServerError-Nachricht ~~antworten~~ mit der Fehlermeldung "Counter too small" antworten.

Anderenfalls (also Zählerwert ist OK) MUSS der Server seine Datenbasis in Bezug auf den mit der KeyID verbundenen Zähler aktualisieren, und der Server verarbeitet die Plaintext-Daten weiter.

Der Server erzeugt zunächst die Datenstruktur "P2" mit

```
 P2=Version (ein Byte mit dem Wert 0x01) ||
    Zählerwert aus dem Request + 1 (unsigned 64-Bit, big-endian-format) ||
    Anzahl der Bytes der folgenden optionalen zusätzlichen HTTP-Header-Informationen (unsigned 32-Bit im Big-Endian-Format) ||
    optionale HTTP-Header-Informationen ||
    Klartext-Antwort des Servers
```

Falls es zu einen Zählerüberlauf kommt, so MUSS der Server eine VAUServerError-Nachricht mit der Fehlermeldung "counter overflow" senden, die KeyID in dessen Datenbasis verwerfen, die damit verbundenen Schlüssel sicher löschen und die Applikationsdaten ("Klar-

text-Antwort des Servers") verwerfen.

Der Server MUSS P2 mit AES-256-GCM verschlüsseln. Dafür MUSS der Server zufällig eine 96-Bit-Nonce erzeugen und bei der AES-GCM-Verschlüsselung verwenden. Die zu übermittelnde Datenstruktur MUSS folgende Form besitzen

```
256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag
```

Diese Datenstruktur MUSS der Server per HTTP-Response mit Content-Type 'application/octet-stream' ohne weitere Kodierungen versenden.
[<=]

[… die folgenden Anforderungen in Abschnitt 6.8 bleiben unverändert …]

6.11 VAU-Kanal und MTOM/XOP

Nachdem die Etablierung des VAU-Kanals abgeschlossen ist, kommuniziert ein VAU-Client (bspw. ein ePA-FdV) mit der VAU bspw., um einen großen verschlüsselten Arztbrief der Akte hinzuzufügen. Dabei ist es für die Performanz (also auch für die Nutzerakzeptanz) günstig, größere Daten mittels der Kodierung gemäß „SOAP Message Transmission Optimization Mechanism (MTOM)"/„XML-binary Optimized Packaging (XOP)" zu transportieren. MTOM/XOP ist die nach W3C empfohlene Methode, um über HTTP/SOAP größere binäre Daten zu transportieren. Dabei werden die Binärdaten nicht Base64- innerhalb einer XML-Datenstruktur kodiert, sondern beim HTTP/SOAP-Request (bzw. bei der HTTP/SOAP-Response) über eine MIME-Multipart-Kodierung auf HTTP-Ebene. Dabei werden innerhalb der SOAP-Nachricht die Binärdaten über eine ID (<xop:Include href="cid:BeispielID1"/>) referenziert und auf HTTP-Ebene über die „Content-ID" (vgl. das folgende Beispiel) und eine MIME-Kodierung binär kodiert. Dies führt zur Reduktion der Datengröße der zu übertragenden Daten von rund 27,5%.

Beispiel:

Ohne MTOM/XOP:

```
POST /URL1 HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: …

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header />
  <SOAP-ENV:Body>
    <ns1:uploadFileRequest xmlns:ns1="urn:example:Upload">
      <ns1:name>Test-Bild.png</ns1:name>
      <ns1:content> … Hier kommen in Base64 kodierte Daten …
</ns1:content>
    </ns1:uploadFileRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Mit MTOM/XOP:

```
POST /URL1 HTTP/1.1
Content-Type: Multipart/Related; boundary="--=_Part_1_1234--";
type="application/xop+xml"; start="Eintrag-1"; start-
info="text/xml"
Content-Length: …

----=_Part_1_1234--
Content-Type: application/xop+xml; type="text/xml"; char-
set="UTF-8"
Content-Transfer-Encoding: 8bit
Content-ID: Eintrag-1

<s11:Envelope
xmlns:s11='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xmime='http://www.w3.org/2005/05/xmlmime'>
 <s11:Body>
  <m:data xmlns:m='http://example.org/stuff'>
  <m:photo xmime:contentType='image/png'>
   <xop:Include
xmlns:xop='http://www.w3.org/2004/08/xop/include'
        href='cid:Bild-1'/>
</m:photo>
</m:data>
 </s11:Body>
</s11:Envelope></SOAP-ENV:Envelope>

----=_Part_1_1234--
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: Bild-1

… hier kommen die binären Daten …
----=_Part_1_1234--
```

Für die Dekodierung einer solchen MIME-Multipart-Kodierung ist es sehr hilfreich (und ggf. notwendig), als Empfänger den vom Sender intendierten (vollständigen) „Content-Type“ aus dem HTTP-Request-Header bzw. dem HTTP-Response-Header zu kennen. Dieser wird jedoch durch das VAU-Protokoll überschrieben (vgl. A_16945 und A_16952). Um den u. a. nach [gemSpec_FM_ePA#A_16220] und [gemSpec_Frontend_Vers#A_16222] geforderte Verwendung von MTOM/XOP zu unterstützen, wird der originär intendierte Content-Type, der im „start“-Feld ggf. vertrauliche Daten enthalten kann, innerhalb der „optionalen zusätzlichen HTTP-Header-Informationen“ (vgl. A_16945 und A_16952) mitgeliefert.

A_18465 - VAU-Protokoll: MTOM/XOP-HTTP-Header-Informationen

Wenn ein VAU-Client oder ein VAU-Server Übertragungen mittels MTOM/XOP durchführt, so MUSS er die durch MTOM/XOP erzeugten „Content-Type“-Informationen (vgl. [gemSpec_Krypt#6.11 VAU-Kanal und MTOM/XOP]) innerhalb ihrer Nachricht als „zusätzliche HTTP-Header-Informationen“ gemäß A_16945 und A_16952 aufführen.

[<=]

A_18466 - VAU-Protokoll: zusätzliche optionale HTTP-Header-Informationen

Wenn ein VAU-Client oder ein VAU-Server Übertragungen durchführt und in den empfangenen Nachrichten sind „zusätzliche optionale HTTP-Header-Informationen“ gemäß A_16945 und A_16952, so MÜSSEN er diese auswerten (Hinweis: insbesondere „Content-Type“-Informationen (vgl. [gemSpec_Krypt#6.11 VAU-Kanal und MTOM/XOP]).

[<=]