Tinyheb – Zertifikatsbehandlung, Signierung und Verschlüsselung

In diesem Dokument wird die Zertifikatsbehandlung, Signierung und Verschlüsselung der freien Abrechnungssoftware für Hebammen tinyheb kurz beschrieben.

Inhaltsverzeichnis

[Kapitel 1 Einführung 2](#_Toc410222340)

[Kapitel 2 Zertifikatsgenerierung 3](#_Toc410222341)

[Kapitel 3 Zertifikatseinspielung 5](#_Toc410222342)

[Kapitel 4 Verschlüsselung und Signierung von Daten 7](#_Toc410222343)

[Kapitel 5 Sonstiges 9](#_Toc410222344)

[Kapitel 6 Erforderliche Anpassungen SHA256 10](#_Toc410222345)

# Einführung

Es ist für einige Datenannahmestellen erforderlich, dass die zu übertragenden Daten verschlüsselt und ihre Echtheit durch die Signatur mittels eines Zertifikats sichergestellt wird. Der aktuelle Zertifikatsgenerierungsprozess mit dem ITSG Trust Center wird von tinyheb derzeit für den Verschlüsselungsalgorithmus SHA1 unterstützt.

Im Zuge der höheren Sicherheitsanforderungen ist hierfür nun zukünftig mit dem Verschlüsselungsalgorithmus SHA256 zu arbeiten (siehe hierzu <http://www.itsg.de/upload/howto_p10_openssl_3821.txt>). Dabei sind folgende Einstellungen für OPENSSL zwingend erforderlich:

* default\_bits = 2048
* string\_mask = nombstr

Eine grundsätzliche Beschreibung des Zertifikatsgenerierungsprozesses ist unter <http://www.itsg.de/upload/130205_So_erhalten_Sie_Ihr_Zertifikat_als_Leistungserbringer_1213.pdf> zu finden.

Der Anwenderprozess für die Zertifikatsbehandlung in tinyheb ist in der entsprechenden Dokumentation im Kapitel *A.4. Ein eigenes Zertifikat* zu finden und wird daher hier nicht weiter betrachtet.

Es wird nun hier rein die softwaretechnische Ebene betrachtet.

# Zertifikatsgenerierung

Für ein Zertifikatsrequest werden folgende Informationen herangezogen:

* IK Nummer der Hebamme (hinterlegt in der tinyheb Datenbank)
* Name der Hebamme (hinterlegt in der tinyheb Datenbank)
* Name Ansprechpartner (wird individuell eingegeben)
* Passwort für Zertifikat (wird individuell eingegeben)
* Länderkennzeichen (festgelegt auf den unveränderlichen Wert *DE*)
* Organisation des Zertifikats (festgelegt auf den für Hebammen unveränderlichen Wert *ITSG TrustCenter fuer sonstige Leistungserbringer*)

Aus diesen Informationen wird u.a. der DN (distinguished name) als Subjekt für die Ablage im LDAP beim Trust Center und im Zertifikat generiert. Er dient dann als u.a. eindeutiger Suchschlüssel für die Datenverifikation.

Verarbeitung für den bisherigen Algorithmus SHA1

Technischer Ablauf des Zertifikatsrequests in tinyheb

Die nachfolgenden Schritte werden derzeit unter Verwendung der freien Softwareimplementierung OPENSSL für Transport Layer Security (ursprünglich Secure Sockets Layer (SSL)) durchgeführt.

Im ersten Verarbeitungsschritt wird aus dem zuvor eingegebenem Passwort ein privater Schlüssel mittels OPENSSL mittels des Verfahrens DES-EDE3-CBC generiert und in der Datei *privkey.pem* abgespeichert. Der zugehörige OPENSSL Befehl lautet:

* openssl genrsa -passout pass:\"$priv\_pass\" -des3 2048

Wer möchte, kann dann zusätzlich noch sein eingegebenes Passwort verschlüsselt auf der Platte in der Datei *pass.pem* ablegen. Die Verschlüsselung erfolgt ebenfalls mittels OPENSSL mittels des von Thomas Baum gelieferten tinyheb Zertifikats *certs/tinyheb\_cert.pem* (Hinweis: Dieses Zertifikat läuft derzeit am 10.03.2015 ab. Daher ist hier eine entsprechende Klärung erforderlich, da danach sehr wahrscheinlich die verschlüsselte Passwortablage nicht mehr funktioniert.). Der zugehörige OPENSSL Befehl lautet:

* openssl smime -encrypt -out $path/pass.pem -outform DER -des3 ../certs/tinyheb\_cert.pem

Anschließend erfolgt die eigentliche Generierung des Zertifikatsrequests mittels OPENSSL unter Verwendung der bisherigen Daten wie z.B. den privaten Schlüssel und den festgelegten Subject-Informationen. Der zugehörige OPENSSL Befehl lautet:

* openssl req -new -key $path/privkey.pem -passin pass:\"$priv\_pass\" -outform DER -subj \"/C=DE/O=ITSG TrustCenter fuer sonstige Leistungserbringer/OU=$name\_hebamme/OU=$ik/CN=$ansprechpartner/\"

Die Ablage des Zertifikatsrequests erfolgt unter der Datei 8-stellige Hebammen-IK-Nummer mit der Endung „.crq“ in der X509-DER Codierung (Distinguished Encoding Rules). Dabei ist PKCS#10 der Certification Request Standard, beschrieben im RFC 2986 und ist da standardisierte Format der Nachrichten, die zu einer Zertifizierungsstelle (certification authority) gesendet werden, um die Zertifizierung eines Schlüsselpaares zu erfragen. Eine Kontrolle des Zertifikatsrequest kann mittels des folgenden OPENSSL Befehl erfolgen:

* openssl req -inform DER -in $ik.crq -text -verify -noout

Als nächsten Schritt ist die Extraktion des öffentlichen Schlüssels aus dem Zertifikatsrequest erforderlich. Hierzu wird die zuvor geschriebene CRQ-Datei herangezogen und mittels des folgenden OPENSSL Befehl durchgeführt (wobei $out\_path auf die CRQ-Datei verweist):

* openssl req -in $out\_path -inform DER -pubkey -noout

Dieser so erzeugte öffentliche Schlüssel wird in der Datei *pubkey.pem* weggeschrieben.

Als nächsten Schritt erfolgt die Berechnung der MD5 Prüfsumme des öffentlichen Schlüssels aus der Datei *pubkey.pem* mittels des ASN.1 Diagnostic Utility sowie deren Ablage als ASN1 Struktur (allerdings DER Encoded) auf dem Dateisystem in der Datei *pubkey\_itsg.der*. Der zugehörige OPENSSL Befehl lautet:

* openssl asn1parse -in $path/pubkey.pem -offset 24 -length 270 -out $path/pubkey\_itsg.der

Um nun die Prüfsumme als MD5 Hashwert (MD5: kryptographische Hashfunktion Message-Digest Algorithm 5) (wird auch Komprimat des Schlüssels oder Fingerprint genannt) zu erhalten, ist folgender OPENSSL Befehl (digest function) auszuführen:

* openssl dgst -md5 -c $path/pubkey\_itsg.der

Abschließend wird noch die SHA1 Prüfsumme des öffentlichen Schlüssels berechnet, die derzeit im gesamten Zertifikatsprozess keine besondere Bedeutung besitzt. Der zugehörige OPENSSL Befehl lautet:

* openssl dgst -sha1 -c $path/pubkey\_itsg.der

Nachdem dies alles erfolgreich durchlaufen worden ist, werden die entsprechenden Informationen der Zertifikatsanfrage (mit CRQ-Datei im Anhang) per E-Mail an das ITSG Trust Center geschickt. Dabei ist die E-Mail Adresse [crq@itsg-trust.de](mailto:crq@itsg-trust.de) derzeit fest im tinyheb Code hinterlegt. Zusätzlich zu der E-Mail sind weitergehende Informationen per Brief an das ITSG Trust Center zu schicken. Ein entsprechender Musterbrief ist in der tinyheb Dokumentation Kapitel A4 zu finden.

# Zertifikatseinspielung

Sind alle Unterlagen zur Zertifikatsanfrage vollständig und erfolgreich bei dem ITSG Trust Center verarbeitet worden, so verschickt das ITSG Trust Center eine E-Mail an die zuvor verwendete E-Mail Adresse der Zertifikatsanfrage mit dem eigentlichem Zertifikat im PKCS#7 Format der Cryptographic Message Syntax Standard, beschrieben im RFC 2315 bzw. RFC 5652 im Anhang mit dem Dateinamen IK-Nummer-Hebamme.p7c. Innerhalb dieser Datei befindet sich das eigentliche angefragte Zertifikat sowie die eine Liste der aktuell gültigen ITSG Zertifikate für den Datenaustausch im Gesundheits- und Sozialwesen die beiden Unterbereiche Datenaustausch im Gesundheits- und Sozialwesen und ITSG TrustCenter fuer sonstige Leistungserbringer. Entscheidend für die weitere Verarbeitung ist dabei eigentlich nur das eigene Zertifikat.

Zur Vorprüfung kann man sich den Inhalt der angehängten PKCS#7 Datei mit folgenden OPENSSL Befehl anschauen:

* openssl pkcs7 -inform DER -print\_certs -in $IK.p7c

Da die aktuelle Tinyheb Implementierung es noch nicht vorgesehen hat, direkt mit dieser angehängten PKCS#7 Datei zu arbeiten, muss man sich die Datei *gesamt-pkcs.key* vom FTP Server des ITSG Trust Center herunterladen und dies dann entsprechend in tinyheb verarbeiten lassen. Es ist zu beachten, dass die Datei gesamt-pkcs.key täglich von der ITSG neu generiert wird, d.h., dass eigene Zertifikat ist erst einen Tag nach der Bestätigungsmail in der Datei auf dem FTP Server enthalten.

Verarbeitung für den bisherigen Algorithmus SHA1

Technischer Ablauf der Zertifikatsresponse vom ITSG Trust Center in tinyheb

In Tinyheb ist derzeit das Zertifikatsladen in der PERL-Datei xkey.pl implementiert. In einer Aufrufmaske werden die erforderlichen Parameter eingegeben und die entsprechende Date Zeilenweise verarbeitet. Aus jeder Zeile der Eingabedatei wird unter Angabe der erforderlichen Header und Trailer eine temporäre Datei im PEM Format (PEM: Privacy-Enhanced Mail; umschlossen von „-----BEGIN CERTIFICATE-----“ und „-----END CERTIFICATE-----“) angelegt, mit OPENSSL die entsprechenden X509-Parameter (CERT Options) ausgelesen und mit der Eingabe verglichen. Betrachtet werden dabei nur alle Zertifikate des Herausgebers des Schlüssels *ITSG TrustCenter fuer sonstige Leistungserbringer*. Wird die IK Nummer der Hebamme aus dem DN des aktuellen X509 Zertifikats ermittelt, so wird die temporäre Datei in die entsprechende $IK.pem Datei gespeichert. Sie enthält nun alle erforderlichen Informationen des von der Hebamme angeforderten Zertifikats. Der zugehörige OPENSSL Befehl lautet:

* openssl x509 -in $cert\_name -subject -dates -serial -noout -certopt no\_header -certopt no\_subject -certopt no\_sigdump -certopt no\_validity -certopt no\_serial -certopt no\_version -certopt no\_issuer -certopt no\_signame –text

Am Ende befindet sich das angefragte, aus der Datei *gesamt-pkcs.key* extrahierte Zertifikat der Hebamme im Verzeichnis /tinyheb/privkey/IK-Nummer.pem und sollte unbedingt zusätzlich gesichert werden.

Manueller Workaround

Wer dieses Prozedere manuell durchführen möchte, kann mittels folgendem Befehl das Zertifikat aus der vom ITSG Trust Center gelieferten $IK.p7c Datei herausziehen und mittels eines geeignete Editors entsprechend unter /tinyheb/privkey/IK-Nummer.pem (IK-Nummer ist dann 8-stellig aufzufüllen) ablegen und somit für tinyheb zur Verfügung stellen. Der zugehörige OPENSSL Befehl lautet:

* openssl pkcs7 -inform DER -print\_certs -in $IK.p7c > p7certs.txt

Geprüft werden kann das Zertifikat (insbesondere der Signatur Algorithmus) mittel folgenden OPENSSL Befehl:

* openssl x509 –in $IK.pem -text

# Verschlüsselung und Signierung von Daten

In der tinyheb Dokumentation im *Kapitel B.1. Generierung von elektronischen Rechnungen* wird einiges zu diesem Thema erläutert. Alle erforderlichen Funktionen sind in der Datei *xauftrag.pl* implementiert oder werden von hier aus aufgerufen.

Im ersten Schritt wird für das evtl. Signieren von Rechnungsdaten das Passwort des privaten Zertifikats der Hebamme benötigt (Methode *get\_sig\_pass*). Sobald eine Rechnung an eine Krankenkasse (Datenannahmestelle) verschickt werden soll, die dies erfordert, wird dieses Passwort einmalig angefordert (wird über den eindeutigen, in der tinyheb Datenbank hinterlegten Parameter SIG + ZIK-Nummer der Krankenkasse festgemacht; ZIK: zentrale IK, Datenannahmestelle mit Entschlüsselungsbefugnis).

Im weiteren Schritt werden die eigentlichen Nutzdaten der Rechnung im definierten EDIFACT-Format erzeugt und unverschlüsselt in der Datei *test\_enc* im temporären Verzeichnis abgelegt. Anschließend wird der öffentliche Schlüssel der zugehörigen Datenannahmestelle aus der tinyheb Datenbank ausgelesen und in der Datei *zik.pem* (PEM-Zertifikatsformat) im temporären Verzeichnis abgelegt. Wer sich diesen Schlüssel in Klartext ansehen möchte, kann dies mit dem zugehörigen OPENSSL Befehl machen:

* openssl x509 -in zik.pem -text

Im folgenden Schritt werden die Nutzdaten der Rechnung (die hierfür erforderlichen Daten sind in der Datei *test\_enc* abgelegt) signiert, sofern dies für die zugehörige Datenannahmestelle erforderlich ist. Hierzu wird die Methode *sig* aus dem PERL-Modul *Heb\_Edi.pm* verwendet. Innerhalb dieser Methode wird dann über die weitere Methode *get\_cert\_info* die zum eigenen Zertifikat der Hebamme passende ITSG Trust Center Datei mit dem passenden, öffentlichen Schlüssel über eine von tinyheb definierte Zertifizierungskette (numerisch fortlaufend von 0 bis derzeit 5) ermittelt. Dazu werden die entsprechenden Informationen aus dem privaten Zertifikat der Hebamme ermittelt (insbesondere wird hierfür das „gültig von“ (notBefore) herangezogen). Der zugehörige OPENSSL Befehl lautet:

* openssl x509 -in $IK.pem -dates -serial -noout

In der so ermittelten (in tinyheb im CERTS-Verzeichnis hinterlegten) Datei *itsg?.pem* (wobei das ? für die tinyheb Zertifizierungskettennummer steht) befindet sich der zum eigenen Zertifikat der Hebamme passende öffentliche Schlüssel des ITSG Trust Center. Der zugehörige OPENSSL Befehl lautet:

* openssl smime -sign -binary -in $Nutzdatendatei -nodetach -outform DER -signer $IK.pem –certfile $itsg?.pem -passin pass:Passwort -inkey $privkey.pem

Mit diesem Befehl wird das SMIME Utility (SMIME: Secure / Multipurpose Internet Mail Extensions) von OPENSSL zum Signieren (Option -sign) verwendet. Das Signieren der Nutzdaten (EDIFACT Nachricht der Rechnung) erfolgt somit unter Verwendung des Zertifikats der Hebamme als eigentlicher Signierer (Option -signer) und zusätzlichem Hinzufügen des zugehörigen Verifikationszertifikats (Option –certfile) sowie die Verwendung des privaten Schlüssels der Hebamme (Option –inkey) und dem zugehörigen Passwort zur Ermittlung des privaten Schlüssels (Option: -passin). Die so die signierten Nutzdaten der Rechnung werden in der Datei *test\_enc.sig* im temporären Verzeichnis abgelegt. Falls die Nutzdaten nicht signiert wurden, ist diese Datei identisch zur Datei *test\_enc*. im temporären Verzeichnis. Wer sich die signierten Daten im ASN.1 Format ansehen möchte, kann dies mit dem zugehörigen OPENSSL Befehl machen:

* openssl asn1parse -in test\_enc.sig -inform DER -dump

Im folgenden Schritt werden die signierten oder unsignierten Nutzdaten der Rechnung (die hierfür erforderlichen Daten sind in der Datei *test\_enc.sig* abgelegt) mit dem öffentlichen Schlüssel der Datenannahmestelle (hinterlegt in der Datei *zik.pem* im temporären Verzeichnis) verschlüsselt, sofern dies für die zugehörige Datenannahmestelle erforderlich ist (wird über den eindeutigen, in der tinyheb Datenbank hinterlegten Parameter SCHL + ZIK-Nummer der Krankenkasse festgemacht; ZIK: zentrale IK, Datenannahmestelle mit Entschlüsselungsbefugnis). Hierzu wird die Methode *enc* aus dem PERL-Modul *Heb\_Edi.pm* verwendet. Der zugehörige OPENSSL Befehl lautet:

* openssl smime -encrypt -binary -in $SigNutzdatendatei -des3 -outform DER $zik.pem

Mit diesem Befehl wird das SMIME Utility (SMIME: Secure / Multipurpose Internet Mail Extensions) von OPENSSL zum Verschlüsseln (Option -enc) verwendet. Das Verschlüsseln der signierten oder unsignierten Nutzdaten erfolgt somit unter Verwendung des öffentlichen Schlüssels der Datenannahmestelle. Die so verschlüsselten Daten werden in der Datei *test\_enc.enc* im temporären Verzeichnis abgelegt. Falls die Nutzdaten nicht signiert wurden, ist diese Datei identisch zur Datei *test\_enc.sig*. im temporären Verzeichnis. Wer sich die signierten Daten im ASN.1 Format ansehen möchte, kann dies mit dem zugehörigen OPENSSL Befehl machen:

* openssl asn1parse -in test\_enc.enc -inform DER -dump

An dieser Stelle ist der erste Teil (der gesamte Zertifikats- und Verschlüsselungsteil) der Rechnungsgenerierung abgeschlossen. Im weiteren Verlauf erfolgt lediglich die Berücksichtigung der Datenaustauschreferenzen sowie die Betriebsphase der Datenannahmestelle, Versendung der Rechnung per Mail mit den entsprechenden Anhängen sowie die Archivierung der generierten temporären Datei in entsprechenden Unterverzeichnissen des temporären Verzeichnisses. Eine weitere Betrachtung dieser Abläufe ist an dieser Stelle nicht erforderlich. Es wird hiermit auf die tinyheb Dokumentation im *Kapitel B.1. Generierung von elektronischen Rechnungen* für die weitergehenden Informationen verwiesen.

# Sonstiges

In diesem Kapitel werden die sonstigen, in tinyheb implementierten Funktionalitäten in Hinblick auf Schlüsselverwaltung und Zertifikatsverwaltung betrachtet. Dazu zählen unter anderem folgende Funktionen:

* Verarbeitung der Kostenträgerdateien
* Verarbeitung der öffentlichen Schlüssel der Datenannahmestellen

Verarbeitung der Kostenträgerdateien

Im PERL-Modul *kostentraeger.pl* erfolgt die Verarbeitung der Kostenträgerdateien, die in einem definiertem EDIFACT-Format entsprechend im Internet online zur Verfügung gestellt und quartalsweise aktualisiert werden. Der entsprechende Ablauf ist in der tinyheb Dokumentation im Kapitel *A.2.1 Verarbeitung der Kostenträgerdateien* entsprechend beschrieben. Bei der Einspielung werden keine OPENSSL Funktionalitäten verwendet. Die entsprechende Kostenträgerdatei wird in seine einzelnen Krankenkassenbestandteile zerlegt und die Daten seriell verarbeitet in der tinyheb Datenbank abgespeichert bzw. aktualisiert.

Verarbeitung der öffentlichen Schlüssel der Datenannahmestellen

Die öffentlichen Schlüssel der Datenannahmestellen werden vom ITSG Trust Center entsprechend im Internet online zur Verfügung gestellt und sowie im Bedarfsfalle aktualisiert werden. Im PERL-Modul *key.pl* erfolgt die Verarbeitung der öffentlichen Schlüssel der Datenannahmestellen. Der entsprechende Ablauf ist in der tinyheb Dokumentation im Kapitel *A.2.2 Verarbeitung der öffentlichen Schlüssel der Datenannahmestellen* entsprechend beschrieben. Die entsprechende Schlüsseldatei mit den einzelnen öffentlichen Schlüsseln der Datenannahmestellen wird in seine einzelnen Datenannahmestellenbestandteile zerlegt und die Daten seriell verarbeitet. Aus jeder Zeile der Eingabedatei wird unter Angabe der erforderlichen Header und Trailer eine temporäre Datei im PEM Format (PEM: Privacy-Enhanced Mail; umschlossen von „-----BEGIN CERTIFICATE-----“ und „-----END CERTIFICATE-----“) angelegt, mit OPENSSL die entsprechenden X509-Parameter (CERT Options) ausgelesen und die einzelnen Parameter ermittelt. Der zugehörige OPENSSL Befehl lautet:

* openssl x509 -in $cert\_name -subject -dates -serial -noout -certopt no\_header -certopt no\_subject -certopt no\_sigdump -certopt no\_validity -certopt no\_serial -certopt no\_version -certopt no\_issuer -certopt no\_signame –text

Die so ermittelten Daten werden seriell verarbeitet und in tinyheb in der Maske angezeigt. Eine Aktualisierung des öffentlichen Schlüssels der Datenannahmestelle erfolgt mittels der Methode *krankenkassen\_up\_pubkey* aus dem PERL-Modul *Heb\_Krankenkasse.pm*. In der tinyheb Datenbank ist dieser Wert im Feld *PUBKEY* der Tabelle *Krankenkassen* unter dem Primärschlüsselfeld *IK* (extrahiert aus dem DN *OU=IK…* des Zertifikats) abgespeichert. Eine weitergehende Betrachtung des verwendeten Algorithmus erfolgt an dieser Stelle nicht.

# Erforderliche Anpassungen SHA256

**Hinweis:**

**Alle hier aufgeführten Informationen basieren lediglich auf meinen persönlichen Erkenntnissen bei der Durchsicht der nicht von mir selbst entwickelten tinyheb Sourcen für diesen besonderen Betrachtungsfall. Das von meiner Frau in tinyheb verwendete Zertifikat wurde mit dem Algorithmus SHA1 erstellt und läuft noch bis zum 31. Juli 2016, sodass wir derzeit noch kein neues Zertifikat mit dem Algorithmus SHA256 beantragen müssen. Derzeit sind mir auch von Seiten meiner Frau keinerlei Probleme bei Signieren und Verschlüsseln von Rechnungsdaten bekannt. Tinyheb läuft bei uns seit vielen Jahren fast unverändert (bis auf die bekannten Patches) unter Windows VISTA.**

In diesem Kapitel erfolgt eine Kurzbetrachtung der aus meiner Sicht erforderlichen Anpassungen bei der Verwendung des neu geforderten Algorithmus SHA256 (SHA: Secure Hash Algorithm) bei der Verwendung der eigenen Zertifikate. Davon betroffen sind daher nur folgende Teilfunktionalitäten von tinyheb:

* Erstellung eines Zertifikatsrequests
* Signierung von Nutzdaten

Erstellung eines Zertifikatsrequests

Die Erstellung eines Zertifikats besteht aus unterschiedlichen Phasen.

Zuerst ist die Generierung eines privaten Schlüssels mit einen eigenem Passwort erforderlich, Hier werden keine SHA256 Funktionalitäten gebraucht, weil hierzu der DES3 Algorithmus zum Verschlüsseln verwendet wird und kein Hashwert benötigt wird.

Die eigentliche Generierung des Zertifikatsrequests erfolgt mittels OPENSSL unter Verwendung der bisherigen Daten wie z.B. den privaten Schlüssel und den festgelegten Subject-Informationen. Der zugehörige OPENSSL Befehl für SHA256 lautet:

* openssl req -new –sha256 -key $path/privkey.pem -passin pass:\"$priv\_pass\" -outform DER -subj \"/C=DE/O=ITSG TrustCenter fuer sonstige Leistungserbringer/OU=$name\_hebamme/OU=$ik/CN=$ansprechpartner/\"

Der so generierte Zertifikatsrequests ist dann weiterhin im PKSC#10 Format, allerdings nun mit dem Signature Algorithm sha256WithRSAEncryption.

Die weitere Berechnung der Prüfsumme MD5 ist vom verwendeten Signaturalgorithmus unabhängig.

Neu wird nun zusätzlich die SHA256 Prüfsumme des öffentlichen Schlüssels (früher: SHA1 Prüfsumme) berechnet, die derzeit im gesamten Zertifikatsprozess keine besondere Bedeutung besitzt, aber in der Implementierungsanleitung für OPENSSL vom ITSG Trust Center erwähnt wird. Ob auch noch weiterhin die SHA1 Prüfsumme benötigt wird, ist noch mit dem ITSG Trust Center zu klären (wahrscheinlich nicht, da die MD5 Prüfsumme bevorzugt wird). Der zugehörige OPENSSL Befehl lautet:

* openssl dgst –sha256 -c $path/pubkey\_itsg.der

Die so generierten Daten können nun wie bisher an das ITSG Trust Center geschickt werden. Die Dateiendung muss nicht zwingend .p10 sein, sondern darf auch weiterhin .crq heißen. Wichtig ist der Inhalt der Datei, der dem geforderten PKCS#10 Format entsprechen muss und nun einen mit SHA256 generierten Zertifikatsrequest enthalten muss.

Irgendwann wird man dann vom ITSG Trust Center eine Rückantwort mit einer Datei der Endung .p7c (PKCS#7 Format) enthalten. In dieser Datei befindet sich neben dem eigentlichen Zertifikat u.a. auch der für die Signierung zu verwendenden Verifikationszertifikat des ITSG Trust Center (zum eigenem Zertifikat gültiger öffentlicher Schlüssel des ITSG Trust Centers zur weiteren Verifikation). Dieser gelieferte ITSG Trust Center Schlüssel ist dann meiner Meinung nach bei der Signierung unbedingt zu verwenden. Dies sollte auch so durch die bisher in tinyheb implementierte Zertifizierungskette gewährleistet sein und führt daher wohl nun mit der SHA256 Einführung zu Problemen.

Signierung von Nutzdaten

Für die Umstellung auf den SHA256 Algorithmus sind meiner Meinung nach keinerlei Anpassungen an den bisher in tinyheb verwendeten OPENSSL Befehlen vorzunehmen. Entscheidend ist hier die richtige Wahl des zu verwendenden öffentlichen Schlüssels des ITSG Trust Center als zertifikatsaustellende Stelle für das eigene Zertifikat der Hebamme. Das bisher implementierter Verfahren der Methode *get\_cert\_info* aus dem PERL-Modul *Heb\_Edi.pm* funktioniert meiner Meinung nach so nicht mehr für den aktuellen Doppelbetrieb von SHA1- und SHA256-Zertifikaten.

Es muss also auf jeden Fall hier eine entsprechende Fallunterscheidung für den zu verwendenden Signaturalgorithmus SHA1 und SHA256 gemacht werden. Nach meiner Analyse sieht die derzeitige Zertifizierungskette des ITSG Trust Center für die Organisation *ITSG TrustCenter fuer sonstige Leistungserbringer* wie folgt aus:

* Seriennummer 21 hex (CA\_21.pem) mit dem SHA1 Fingerprint bf c5 e5 4c ec 55 3b 6f ff 9f e0 f2 4f 34 68 83 09 84 95 3c für ***SHA1 Zertifikate***, die vom 06.12.2011 und bis zum 04.12.2013 erstellt worden sind
* Seriennummer 23 hex (CA\_23.pem) mit dem SHA1 Fingerprint 88 9c ca 00 d7 df 06 45 6d 53 2c 04 96 05 57 42 5d 40 36 1f für ***SHA1 Zertifikate***, die vom 04.12.2013 und bis heute erstellt worden sind
* Seriennummer 2B hex (CA\_2B.pem) mit dem SHA1 Fingerprint d2 5d 68 3c e6 7d 7e 5e 49 46 4e e1 da 10 7d 1f 7d 13 c6 90 für ***SHA256 Zertifikate***, die vom 23.11.2011 und bis zum 04.12.2013 erstellt worden sind
* Seriennummer 2D hex (CA\_2D.pem) mit dem SHA1 Fingerprint 8f f9 54 0e ad cc cb c7 b3 23 ce d1 e8 9a 3f 6f b8 0e 84 d5 für ***SHA256 Zertifikate***, die vom 04.12.2013 und bis heute erstellt worden sind

Wenn man meiner Meinung nach nicht den richtigen, zum eigenem Zertifikat gehörenden ITSG Trust Center Schlüssel beim Signieren der Nutzdaten verwendet, kommt es wohl zu der bereits in der tinyheb Mailingliste diskutierte „Verify error:certificate signature failure“.