Institut für Qualitätssicherung und
Transparenz im Gesundheitswesen

Packer

Anwenderhandbuch V4.2.14

Erstellt im Auftrag des
Gemeinsamen Bundesausschusses

Stand: 07. Januar 2022

Impressum

Thema:
Anwenderhandbuch Packer V4.2.14

Auftraggeber:
Gemeinsamer Bundesausschuss

Datum der Abgabe:
07. Januar 2022

Herausgeber:
IQTIG – Institut für Qualitätssicherung
und Transparenz im Gesundheitswesen

Katharina-Heinroth-Ufer 1
10787 Berlin

Telefon: (030) 58 58 26-0
Telefax: (030) 58 58 26-999

info@iqtig.org
http://www.iqtig.org

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Informationen zu diesem Dokument

Darstellungsmittel

Im Folgenden sind Symbole und Darstellung besonderer Informationen beschrieben.

Achtung

Beschreibt Ursache, Folge und Vermeidung einer besonderen Fehlanwendung, die zu Problemen bei der Implementierung oder Ähnlichem führen kann.

Zielgruppe

Dieses Handbuch richtet sich an administrative Mitarbeiter sämtlicher am Datenfluss beteiligter Stellen und Softwareentwickler, die mit der Umsetzung der XML-Datenflüsse und der damit verbundenen Verschlüsselung einzelner XML-Elemente sowie vollständiger XML-Dateien beschäftigt sind.

Änderungen in der Version 4.2.14

- Packer kann wieder mit Java ab Version 8 betrieben werden.

Änderungen in der Version 4.2.13

- Upgrade der Log4j Dependency Version auf 2.17.0

Änderungen in der Version 4.2.12

- Upgrade der Log4j Dependency Version auf 2.16.0
- Upgrade der BouncyCastle Dependency Version auf 1.70

Änderungen in der Version 4.2.11

- Bugfix: TPacker: Eine Entschlüsselung war nicht möglich, wenn das Passwort Sonderzeichen enthielt und die Verschlüsselungsumgebung (JVM auf dem Betriebssystem) eine andere Standardzeichencodierung als die Entschlüsselungsumgebung nutzte. Grund dafür war, dass der TPacker bis dato immer das Passwort mit der Zeichencodierung der Umgebung interpretierte (JVM-file.encoding).
  - **Novum: Verschlüsselung**: Das Passwort darf fortan nur Zeichen aus folgender Zeichenmenge beinhalten: `a-zA-Z0-9.,:;=+*#_!$%?@(){}|'^~`
  - **Novum: Allgemein**: Der TPacker interpretiert das Passwort immer einheitlich und plattformunabhängig: Ausnahme: siehe nächster Punkt.
  - **Novum: Entschlüsselung**: Es existieren folgende Lösungswege, sofern mit dem TPacker Altdaten entschlüsselt werden müssen, die mit einer früheren Version des TPackers, mit einem Sonderzeichenpasswort, auf einer Plattform mit abweichendem Standard-Encoding, verschlüsselt wurden:
    - Konsolenanwendung: Für die selten abzusehenden Problemfälle wurde für die Entschlüsselung der optionale Parameter –c (--characterset_pw_decrypt) eingeführt, mit dem die Zeichencodierung des Passworts zur Entschlüsselung festgelegt werden kann.

  - API: Der entsprechende API-Parameter (zu –c auf der Konsole) ist dokumentiert unter: org.iqtig.tpacker.Api
  - Sofern ausschließlich mit dem GPacker entpackt wird, kann für die Problemfälle die bisherige Version 4.2.10 behalten werden. Sofern das Problem erstmalig auftritt, muss dieser mit –Dfile.encoding=<Zeichencodierung wie auf der verschlüsselnden Plattform> gestartet werden.

- Anpassungen in diesem Dokument:
  - Absatz zum TPacker angepasst: Neuen optionalen Parameter –c samt Beispielen aufgeführt.

Änderungen in der Version 4.2.10

- Absatz zu Schlüsselformaten hinzugefügt

Änderungen in der Version 4.2.9

- Anmerkung zum Umgang mit Speicherproblemen hinzugefügt.

Änderungen in der Version 4.2.8

- Die Version dieses Handbuchs wird fortan mit dem Versionsstand der Software angegeben, d.h. Version 4.2.8 statt - wie zu erwarten - Version 6.
- API: Library-Updates (gemäß OWASP-Empfehlung): santuario.xmlsec 2.1.3 -> 2.1.4
- Bugfix: TPacker/XPacker: Parameter konnten nicht mehr zusammengerückt werden (bspw. nicht **`-ze`** statt **`-z –e`**, oder nicht **`–zeyt`** statt **`-z -e -y --timestamp`**)

Änderungen in der Version 05

- Bugfix: Die Ausgabe der Hilfe (-h) der Konsolenanwendungen funktionierte nicht.
- Neue Profile wurden dem GPacker hinzugefügt:
  - LE - Leistungserbringer (Datenlieferung mit Transplantationen)
  - LE - Leistungserbringer (Datenlieferung ohne Transplantationen)
- API
  - Die pom.xml-Dateien, die den API-JAR-Dateien beigestellt sind, waren veraltet und unvollständig.
    - Die pom.xml-Dateien führen jetzt alle aktuellen, transitiven Abhängigkeiten der API-JARs auf.
    - Öffentlich verfügbare Abhängigkeiten (bspw. `org.bouncycastle:bcprov-jdk15on`) werden nicht gesondert mit ausgeliefert.
  - Irrelevante Attribute wurden aus `ValidationErrorException` entfernt, bspw. `EnDataStatus:status`.
  - Package-Verschiebungen/Umbenennungen
    - `org.iqtig.packer.util > org.iqtig.packer.shared`
    - `org.iqtig.packer.util.error.ErrorsFromDb.Dechiffrierung > org.iqtig.packer.shared.error.Errors.*`

```
– org.iqtig.packer.util.error.ErrorsFromDb.Verschluesselungsprogramm
    > org.iqtig.packer.shared.error.Errors.*
```

Änderungen in der Version 04

- Grundsätzliche Überarbeitung nach Sanierung des GPackers:
  - zur Unterstützung des XML-Elements *patient_tx*
  - nach Entfernung des XML-Element *feedback_key*
- Entfernung des Kapitels Aktionen
- Übernahme von Informationen aus Textdatei in Kapitel 7 XPacker
- Überarbeitung der Erläuterungen in Kapitel 8 GPacker
- Aktualisierung von Kapitel 9 Anwendungsbeispiele

Änderungen in der Version 03

- Anpassung der Systemvoraussetzungen in Kapitel 2

Änderungen in der Version 02

- Anpassung der Systemvoraussetzungen in Kapitel 2 (JAVA 8)
- Einführung einer Versionierung dieses Anwenderhandbuchs
- Aufnahme des Abschnitts „Informationen zu diesem Dokument“

1 Einleitung

Dieses Anwenderhandbuch dokumentiert die Packer-Anwendungen des IQTIGs. Sie dienen zur Transformation von Dateien im XML-Format zum Austausch im Rahmen von DeQS-RL[1] oder oKFE-RL[2].

Die Anwendungen sind:

1. TPacker – Konsolenanwendung zur Transportverschlüsselung
2. XPacker – Konsolenanwendung zur XML-Inhaltsverschlüsselung
3. GPacker – Grafische Anwendung zur Nutzung von TPacker und XPacker.

2 Voraussetzungen

Die Programme setzen die Installation von Java 8 oder einer höheren Version voraus. Dies gilt insbesondere für die Nutzung von OpenJDK. Ferner werden – je nach Aktion – „private“ oder „öffentliche Schlüssel“ benötigt. Diese werden im folgenden Abschnitt erklärt.

3 Hintergrund der Verschlüsselung

Für die XML Ver- und Entschlüsselung werden sogenannte Schlüssel benötigt. Diese Schlüssel bestehen aus langen Zeichenketten und werden daher in Dateien abgelegt. Da nun mehrere Organisationen diese Dateien, aber nicht jede Organisation alle Daten darin sehen/lesen darf, werden Schlüsselpaare eingesetzt. Jedes Schlüsselpaar besteht aus zwei unterschiedlichen Typen von Schlüsseln:

- Private Schlüssel:
  Der eigene private Schlüssel darf niemals dem Kommunikationspartner gegeben werden.
- Öffentlicher Schlüssel:
  Nur der öffentliche Schlüssel darf dem Kommunikationspartner gegeben werden.

Wenn nun Daten des Absenders A für verschiedene Empfänger (E_1, E_2, usw.) verschlüsselt werden sollen, so benötigt man die öffentlichen Schlüssel von E_1 und E_2 zum Verschlüsseln der Daten von A: Der Bereich der nur für E_1 ist, wird mit dessen öffentlichem Schlüssel chiffriert. Analog mit dem Bereich für E_2 usw.

Wenn nun E_1 die Daten erhält, kann E_1 mit seinem eigenen privaten Schlüssel die für ihn bestimmten Daten entschlüsseln. Er kann nicht die für E_2 bestimmten Daten entschlüsseln.

[1] Richtlinie zur datengestützten einrichtungsübergreifenden Qualitätssicherung
[2] Richtlinie für organisierte Krebsfrüherkennungsprogramme

Dieses Verfahren wird auch in der Qualitätssicherung eingesetzt. Die Leistungserbringer verschlüsseln Daten mit verschiedenen öffentlichen Schlüsseln der Empfänger, welche dann nur die für sie bestimmten Daten entschlüsseln und verarbeiten können.

Abbildung 1: Ver- und Entschlüsselung sowie Ver- und Entpacken der XML-Elemente

Wichtige Hinweise

- Beim Verschlüsseln nur den öffentlichen Schlüssel des Empfängers einsetzen
- Beim Entschlüsseln nur den eigenen privaten Schlüssel einsetzen
- ***Niemals den eigenen privaten Schlüssel an Dritte weitergeben!***
- Nur öffentliche Schlüssel an Dritte geben
- Die Schlüssel werden folgendermaßen bezeichnet:
  - Privater Schlüssel: ' Englisch: Private Key ' Dateiendung: *.pri
  - Öffentlicher Schlüssel: ' Englisch: Public Key 'Dateiendung: *.pub

Die Bereiche, die unterschiedlich verschlüsselt werden müssen, werden auch als "Elemente" in der Anwendung bezeichnet (siehe

Abbildung 4).

Das IQTIG sammelt die öffentlichen Schlüssel der Datenservices der Beteiligten Institutionen und stellt diese auf der Website des IQTIG unter folgender Adresse zur Verfügung: https://iqtig.org/datenerfassung/servicedateien/

Bezüglich der Maßgabe die privaten Schlüssel niemals an Dritte weiterzugeben sei ergänzend angemerkt, dass die XML-Verschlüsselung zur Einhaltung des Datenschutzes erforderlich ist und die Schlüssel ein Hilfsmittel für deren Sicherstellung darstellen. Daher ist insbesondere für den privaten Schlüssel besondere Sorgsamkeit geboten. Dessen Ablage muss gesichert sein und er darf lediglich autorisierten Personen zugänglich sein. Idealer Weise ist der Zugang ohne Passphrase nicht möglich. Für die sichere Ablage kann beispielsweise auf eine Smartcard zurückgegriffen werden. Sollte der private Schlüssel trotz sorgsamem Umgang kompromittiert sein, ist dies unverzüglich nach Kenntnisnahme dem IQTIG mitzuteilen, um einen geregelten, kurzfristigen Schlüsseltausch (insbesondere des öffentlichen Schlüssels) zu initiieren.

3.1 Schlüsselformate

Werden RSA Schlüsselpaare über die API des XPacker erzeugt, so werden der öffentliche Schlüssel und der private Schlüssel jeweils Base64-kodiert und als Dateien (*.pri, *.pub) im Dateisystem abgelegt. Es können allerdings auch RSA Schlüssel verwendet werden, die nicht über den XPacker erzeugt wurden. Diese liegen sehr häufig im PEM-Format vor. Bis zur Version 4.2.9 (einschließlich) des Packers kommt es zu einem Fehler, da die Software mit diesem Format nicht umgehen kann. Ab der Version 4.2.10 können Schlüssel ist dies möglich. Das PEM-Format kennzeichnet sich durch einen Header und Footer, die jeweils mit 5 Spiegelstrichen beginnen und enden.

Beispiel: Public Key (PEM Format)

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqkiG9w0……DfTL721LfXCi3/Tw/b7MIgjn8rQxrQ8M3/tBAPHxr/W7
-----END PUBLIC KEY-----
```

4 Datei-Formate und Prozesse

Grundsätzlich handelt es sich um XML-Dateien, welche vom Leistungserbringer erzeugt werden. Bevor diese jedoch zur Datenannahmestelle transportiert werden können, müssen sie aus Datenschutzgründen bearbeitet werden:

- Zum einen wird in den Dateien eine Verschlüsselung vorgenommen. Verschiedene Bereiche/Elemente werden – wie zuvor beschrieben – mit den öffentlichen Schlüsseln der Empfänger verschlüsselt. Obwohl sich alle Informationen für alle Beteiligten innerhalb einer XML-Datei befinden, ist sichergestellt, dass Unbefugte diese Bereiche/Elemente nicht einsehen können – wohl aber der Empfänger, der die Dateien mit seinem privaten Schlüssel entschlüsselt. Dieser Schritt ist nur notwendig, wenn in der Datei dokumentierte QS-Fälle mit patientenidentifizieren-den Daten (PID) enthalten sind.

Achtung

- Sind keine QS-Daten in der Datei enthalten (wie z.B. bei Protokollen), kann dieser Schritt übersprungen werden.
- Vor und nach der Verschlüsselung/Entschlüsselung handelt es sich immer um XML-Dateien.

- Zusätzlich wird die Datei noch für den Transport per Email „verpackt“. Hierbei wird die Gesamtdatei mit einem symmetrischen Verfahren verschlüsselt. Dazu werden die Registrierungsnummer und das Transport-Passwort benötigt, welches der Absender im Zuge der Registrierung beim Empfänger erhalten hat.

Achtung

- Dieser Schritt ist immer auszuführen, wenn Dateien per Internet oder auf anderen unsicheren Kanälen transportiert werden.
- Die verpackten Dateien werden im binären AES-Format gespeichert, welches eine sichere Verpackung auf unsicheren Wegen darstellt.

---

5 Allgemeines zu Java-Programmen

Kommt es bei Ausführung von Java-Programme zu einem `java.lang.OutOfMemoryError`, so reicht der per Standard allokierte Arbeitsspeicher zur Verarbeitung des Java-Programms nicht aus. Die etablierten Parameter[3][4] zur Steuerung hierzu sind:

- `-Xms<size> set initial Java heap size`
- `-Xmx<size> set maximum Java heap size`

Bspw.:

- `java -jar *Packer*.jar <Parameter> -Xmx8192m`

6 TPacker

Der TPacker ist die Konsolenanwendung für die Anwendung der Transportverschlüsselung. Der Programmaufruf erfolgt mittels folgender Syntax:

```
java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar <Parameter>
```

Parameter:

`-h` gibt die Hilfe aus.

`-z` Die Eingangsdateien werden gepackt. Es wird die Erweiterung ".zip" an die Ausgabedatei angehängt.

`-u` Die Eingangsdateien werden entpackt.

`-e` Die Eingangsdateien werden verschlüsselt. Es wird die Erweiterung ".aes" (symmetrisch) oder ".rsa" (asymmetrisch) an die Ausgabedatei angehängt.

`-d` Die Eingangsdateien werden entschlüsselt. Dabei wird die Erweiterung ".aes" (symmetrisch) oder ".rsa" (asymmetrisch) im Dateinamen der Ausgabedatei entfernt.

`-f` Ent- und Verschlüsselung mehrerer Eingangsdateien, die mit Kommata ohne Leerschritt getrennt werden müssen.

`-o` Pfad oder Dateiname für die Ausgabe. Falls keine Ausgabedatei angegeben wird, wird der erste Name der Eingangsdatei verwendet.

`-p` Passwort für die symmetrische Ver-/Entschlüsselung.

`-c` Optionale Angabe zur Zeichencodierung des Passworts zur Entschlüsselung, bspw. `UTF-8` oder `ISO-8859-1`. Nur gedacht zur Lösung von Entschlüsselungsproblemen

[3] https://docs.oracle.com/en/java/javase/14/docs/specs/man/java.html#extra-options-for-java
[4] https://docs.oracle.com/javase/8/docs/technotes/tools/windows/java.html#BABHDABI

mit Altdateien (verschlüsselt mit Version < 4.2.11) in Kombination mit Sonderzeichen in Passwörtern.

`-t` Wenn diese Option gesetzt ist, wird kein Zeitstempel (TimeStamp) im Ausgabedateinamen erzeugt.

`--unsafe` Überspringt die eingebaute Prüfung, ob `VERSICHERTENID` im Dokument vorhanden ist.

Beispiele für symmetrische Verschlüsselung:

a) Verschlüsseln / Entschlüsseln
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -e -f file -o outpath -p password`
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -d -f file.aes -o outpath -p password`

b) Packen / Entpacken
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -z -f file -o outfile`
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -u -f file.zip -o outpath`

c) Packen und Verschlüsseln
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -ze -f file -o outfile -p password`

d) Entschlüsseln und Entpacken
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -du -f file.zip.aes -o outpath -p password`

e) Entschlüsseln (und Entpacken) von Altdateien, die auf einem System mit Java-Standard-Zeichencodierung ISO-8859-1 (file.encoding=ISO-8859-1) verschlüsselt wurden:
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -d -f file.aes -o outpath -p password –c ISO-8859-1`
   - `java -jar TPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -du -f file.zip.aes -o outpath -p password –c ISO-8859-1`

7 XPacker

Der XPacker ist die Konsolenanwendung für die Anwendung der XML-Verschlüsselung. Der Programmaufruf erfolgt mittels folgender Syntax:

```
java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar [Parameter]
```

Parameter:

`-h` gibt die Hilfe aus.

`-e` Die Eingangsdateien werden verschlüsselt.

`-d` Die Eingangsdateien werden entschlüsselt.

`-g` Es wird ein öffentlicher und ein privater Schlüssel erzeugt.

Standardnamen key.pub, key.pri

`-f` Eingangsdatei, falls keine angegeben wird die Standardeingabe verwendet.

`-o` Ausgabedatei, falls keine angegeben wird alles an die Standardausgabe geleitet.

`-k` Beim Verschlüsseln wird der öffentliche Schlüssel benötigt und

beim Entschlüsseln der private Schlüssel.

`-t` Liste der XML-Tags, die mit dem Schlüssel ver-/entschlüsselt werden.

`-2` XML-Tag an dem der Schlüssel eingebunden werden soll.

Beispiele:

- `java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -e -f inputfile -o outputfile -k keyfile -t tag1,tag2... -2 tagForEncryptedKey`
- `java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -e -k keyfile -t tag1,tag2... -2 tagForEncryptedKey < inputStream > outputStream`
- `java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -d -f inputfile -o outputfile -k privateKeyFile -t tag1,tag2...`
- `java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -d -k keyfile -t tag1,tag2... < inputStream > outputStream`
- `java -jar XPacker-<Version>-SNAPSHOT-jar-with-dependencies.jar -g -k keyname`

8 GPacker

Für alle Anwender, die händisch die Verschlüsselung durchführen müssen, stellt der GPacker mit seiner grafischen Oberfläche eine interaktive Alternative zur Verwendung der Programme XPacker und TPacker dar. Die beiden Programme werden nicht benötigt, wenn der GPacker eingesetzt wird.

Er ermöglicht mit Hilfe einer grafischen Oberfläche die Verschlüsselung und Komprimierung von Dateien im XML-Format, die im Rahmen der DeQS-RL oder oKFE-RL erstellt worden sind.

8.1 Bedienelemente

Die folgende Abbildung 2 zeigt die grafische Benutzeroberfläche, deren Bestandteile im Folgenden näher erläutert werden.

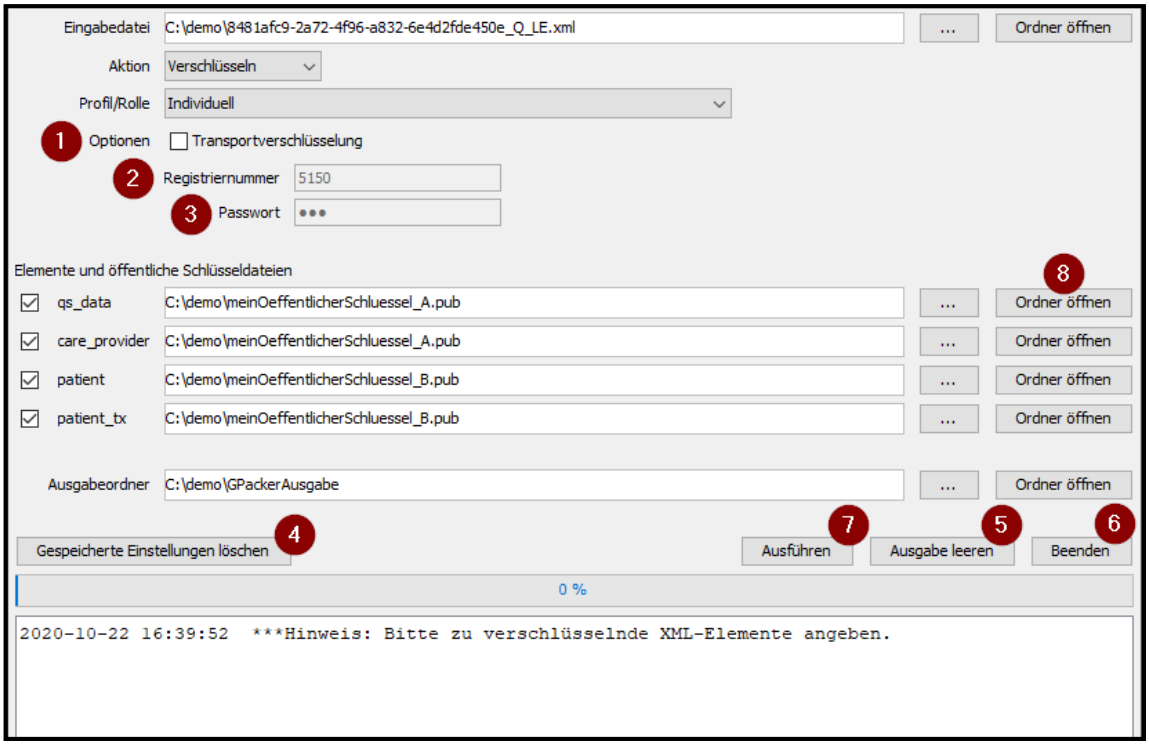

Abbildung 2: GPacker Bedienelemente

1 Checkbox Transportverschlüsselung

- Bestimmt, dass die Transportverschlüsselung im Rahmen der Ausführung behandelt wird.
- Auswählbar im Rahmen der *Aktion=Verschlüsseln*,
    - Die Auswahl führt dann dazu, dass eine zip.aes-Ergebnisdatei erzeugt wird.

- Nicht auswählbar im Rahmen der *Aktion=Entschlüsselung*:
    - Eine automatische Selektion erfolgt aber nach Auswahl einer zip.aes-*Eingabedatei*, denn hier muss als erstes eine Entschlüsselung erfolgen.

2 Textfeld Registriernummer

- Nur aktiv und erforderlich, wenn eine transportverschlüsselte Datei erstellt wird.
- Bestimmt lediglich Teile des Dateinamens der zu erstellenden, transportverschlüsselten Datei und hat darüber hinaus keine steuernde Funktion.

3 Textfeld Passwort

- Bestimmt das Passwort zur *Transportverschlüsselung*.
- Nur aktiv und erforderlich, wenn Transportverschlüsselung im Rahmen der Ausführung behandelt wird, sprich (1) selektiert ist.

4 Schaltfläche Gespeicherte Einstellungen löschen

- Mit diesem Knopf werden alle gespeicherten Einstellungen gelöscht.
- Hinweis: Der GPacker speichert diverse Eingaben für die nächste Benutzung dauerhaft. Beispiel: Ordner der zuletzt gewählten *Eingabedatei*, zuletzt gewählter *Ausgabeordner*, …

5 Schaltfläche Ausgabe leeren

- Löscht den Inhalt des Textbereichs im unteren Teil der Anwendung.

6 Schaltfläche Beenden

- Beendet die GPacker-Anwendung

7 Schaltfläche Ausführen

- Führt die eingegebenen Ver- bzw. Entschlüsselungsanweisungen aus und legt die Ergebnisse in einem Laufordner unterhalb des *Ausgabeordner* ab.
- Fehlen notwendige Eingaben, so wird mittels Fehlermeldung darauf hingewiesen.

8 Schaltfläche Ordner öffnen

- Öffnet den Ordner des Pfads der links nebenstehenden Datei- bzw. Ordnerauswahl.
- Nur aktiv, wenn links nebenstehend eine Datei- bzw. Ordnerauswahl vorgenommen wurde.
- Die Funktionalität aller *Ordner öffnen*-Schaltflächen ist identisch.

8.2 Programmablauf

Die folgenden Szenarien beschreiben den korrekten Ablauf zur Nutzung des GPacker aus Sicht eines Leistungserbringers. Die grafische Benutzeroberfläche nach dem erstem Startvorgang des GPacker wird in Abbildung 3 gezeigt.

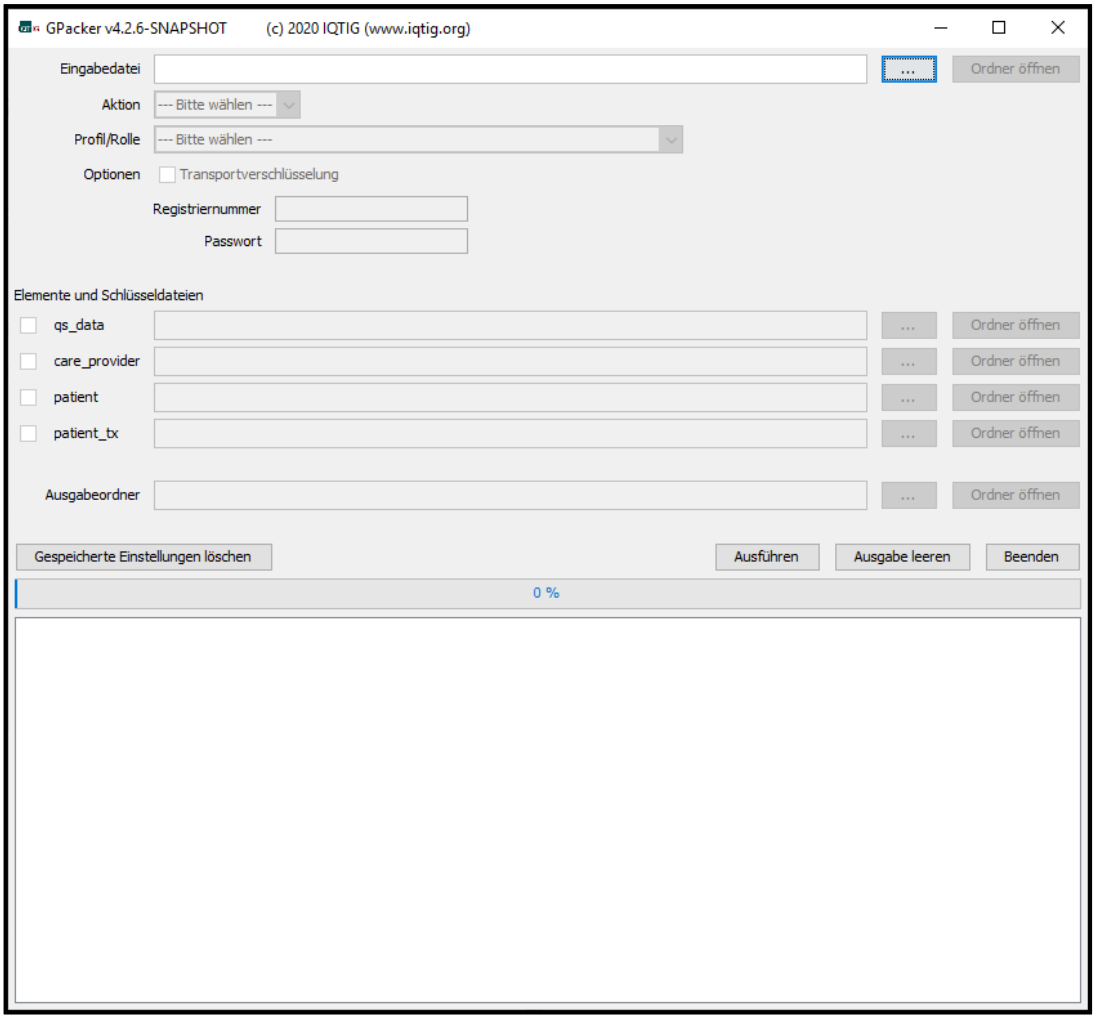

Abbildung 3: GPacker nach dem initialen Start

Szenario 1 – Versand von QS-Daten

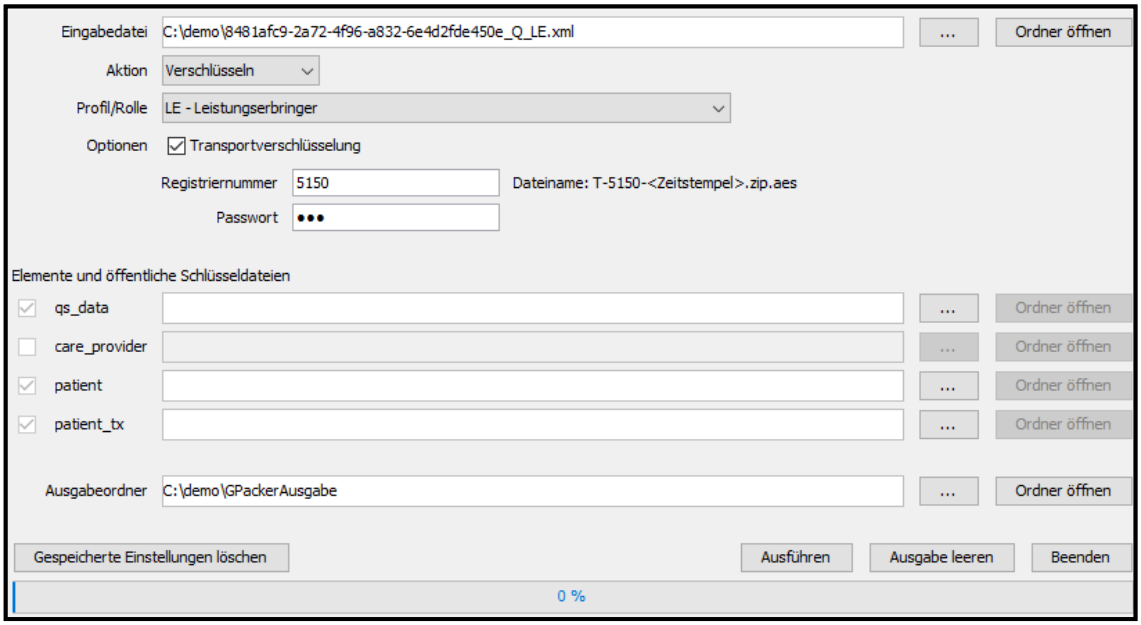

Abbildung 4: Szenario 1 - Versand von QS-Daten

1. Wählen Sie für *Eingabedatei* eine XML-Export-Datei aus, wie sie aus dem QS-Dokumentationsprogramm exportiert wird.
2. Wählen Sie im Drop-down-Menü *Aktion* die Aktion *Verschlüsseln* aus.
3. Wählen Sie im Drop-down-Menü *Profil/Rolle* bspw. das Profil *LE - Leistungserbringer* aus.
4. Selektieren Sie die Checkbox *Transportverschlüsselung*.
5. Setzen Sie im Textfeld *Registriernummer* Ihre Registriernummer ein.
6. Setzen Sie im Textfeld *Passwort* das für Ihre Registrierung bilateral ausgetauschte, geheime Passwort zur Transportverschlüsselung ein.
7. Wählen Sie unter *Elemente und öffentliche Schlüsseldateien* für jedes selektierte Element die zugehörige öffentliche Schlüsseldatei aus. Konkrete Vorgaben sind der jeweiligen zweckgebundenen Spezifikation zu entnehmen. Beispielsweise muss das Element *qs_data* in Abhängigkeit zum Verfahren mit dem öffentlichen Schlüssel entweder der zuständigen Datenannahmestelle auf Landesebene oder der Auswertestelle auf Bundesebene verschlüsselt werden.

   Die Selektion der Elemente wird durch Ihre Auswahl im Drop-down-Menü *Profil/Rolle* bestimmt.

8. Wählen Sie den *Ausgabeordner* über die entsprechende Schaltfläche aus.
9. Klicken Sie auf die Schaltfläche *Ausführen*.

Szenariovariante

Ohne Selektion der Checkbox Transportverschlüsselung werden nur die Elemente der gewählten XML-Datei verschlüsselt, ohne dass dieses Ergebnis wiederum in einer verschlüsselten zip.aes-Datei verpackt wird.

Szenario 2 – Empfang des Rückprotokolls

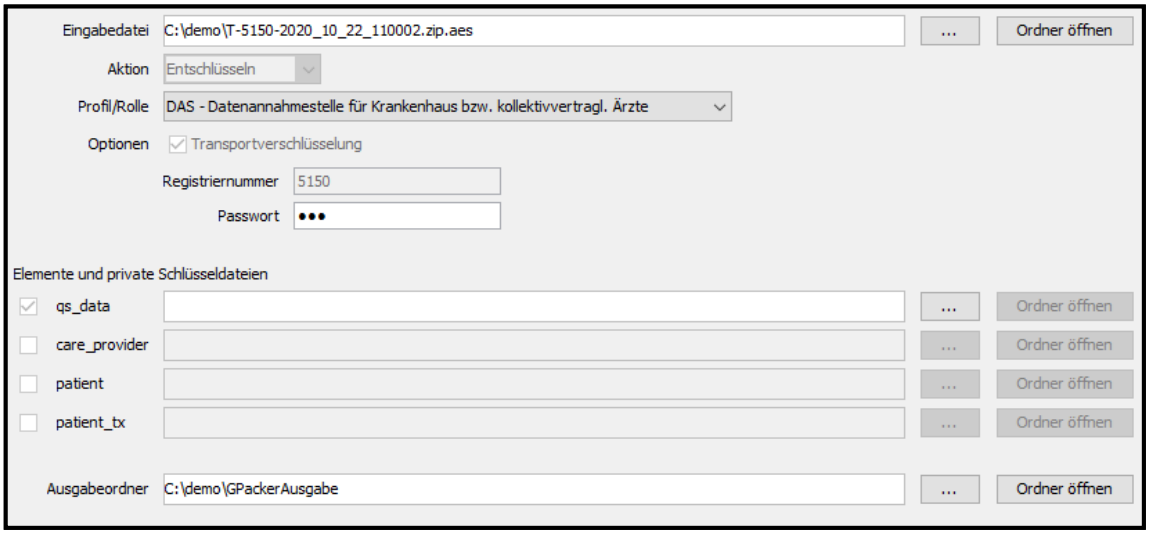

Abbildung 5: Szenario 2 - Empfang des Rückprotokolls

1. Wählen Sie über die Schaltfläche für *Eingabedatei* eine Eingabedatei aus, wie sie von der Datenannahmestelle als Datenflussprotokoll verschickt wird (zip.aes-Datei).
   - Der Wert *Entschlüsseln* wird automatisch im Drop-down-Menü *Aktion* gesetzt.
2. Treffen Sie im Drop-down-Menü *Profil/Rolle* keine Auswahl
   - Es ist lediglich die Aufhebung der Transportverschlüsselung, nicht jedoch einer XML-Entschlüsselung erforderlich, da in Datenflussprotokollen weder das Element *patient* noch das Element *qs_data* enthalten ist.
3. Setzen Sie im Textfeld *Passwort* das für Ihre Registrierung bilateral ausgetauschte, geheime Passwort zur Transportverschlüsselung ein.
4. Wählen Sie den *Ausgabeordner* über die entsprechende Schaltfläche aus.
5. Klicken Sie auf die Schaltfläche *Ausführen*.

Szenariovarianten

1. Wählen Sie unter *Profil/Rolle* das Profil *Individuell* aus, so wird die Transportdatei entschlüsselt, entpackt und es besteht die Möglichkeit die Verschlüsselung einzeln auswählbarer XML-Elemente unter Verwendung des eigenen privaten Schlüssels aufzuheben.
2. Durch Auswahl einer XML-Datei mit verschlüsselten Elementen als Eingabedatei (anstelle der gezeigten zip.aes-Datei) können Sie deren Elemente entschlüsseln.

Wissenswertes

Die angestoßene Entschlüsselung kann, je nach Größe der Datei und Geschwindigkeit Ihres Rechners, mehrere Minuten dauern. Wenn das Entschlüsseln abgeschlossen ist, wird dieses in der Oberfläche dargestellt.

Jede erfolgreiche Ausführung mittels Schaltfläche *Ausführen* erzeugt einen neuen Laufordner.

8.3 Sonderfälle bei Datenlieferungen

Sinn und Zweck, der über das Drop-down-Menü *Profil/Rolle* auswählbaren Profile ist es, die XML-Elemente zur Ver-/Entschlüsselung für absehbare Anwendungsszenarien vorzugeben.

Sollte augenscheinlich kein Profil zu Ihrer Nutzerrolle/Aufgabe passen, so wählen Sie das Profil *Individuell*: Es erlaubt als einziges Profil die freie Auswahl der zu ver-/entschlüsselnden XML-Elemente.

Dieser Expertenmodus ist nur für erfahrende Anwender geeignet.

Das folgende Beispiel in Abbildung 6 zeigt die Auswahl aller verfügbaren Elemente.

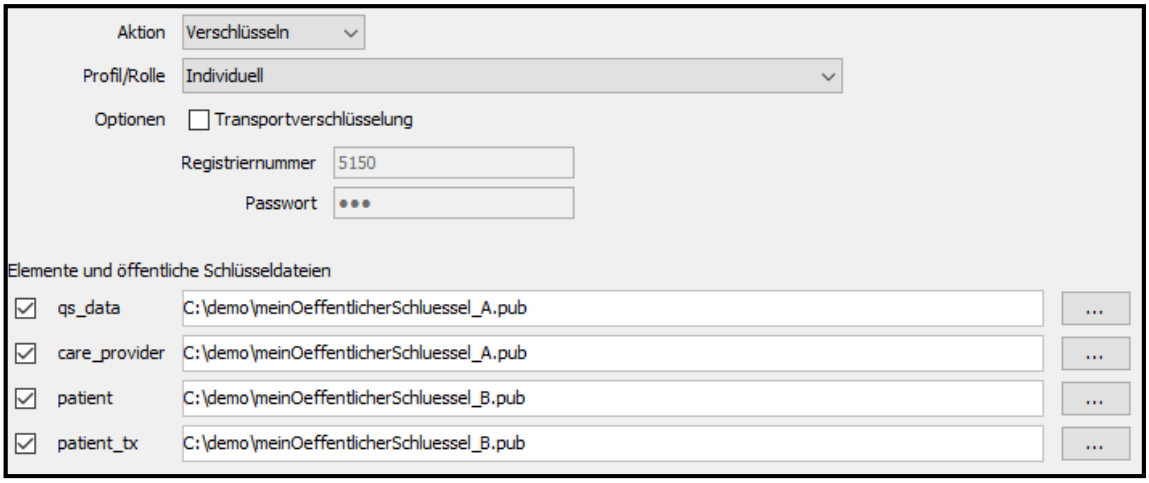

Abbildung 6: Individuelle Auswahl zu verschlüsselnder Elemente

8.4 Fehlermeldungen

Tabelle 1: Fehlermeldungen des GPackers (Auswahl)

Meldung	Erläuterung
Es konnten keine temporären Dateien erstellt werden.	Es wurde kein Ordner zur Erzeugung temporärer Dateien gefunden.
Die Eingabedatei konnte nicht gefunden werden.	Der GPacker konnte die Eingabedabei nicht finden oder nicht öffnen.
Bitte Registriernummer angeben.	Es muss bei dieser Aktion eine Registriernummer angegeben werden. Dieses wurde Ihnen bei der Registrierung mitgeteilt.
Bitte Passwort angeben.	Es muss bei dieser Aktion ein Transport-Passwort angegeben werden. Dieses wurde Ihnen bei der Registrierung mitgeteilt.

9 Anwendungsbeispiele

Die folgenden Anwendungsbeispiele sollen häufige Szenarien aus dem Alltag der Beteiligten im Datenfluss wiedergeben. Die Fälle sind gruppiert nach der Rolle des Anwenders.

9.1 Leistungserbringer (Kliniken/Praxen)

- Die Datenerfassung in der Dokumentationssoftware ist abgeschlossen und die Exportdatei wurde erstellt und soll nun an die Datenannahmestelle verschickt werden.
- Im GPacker muss die XML-Verschlüsselung entsprechend des Datenflusses konfiguriert werden:

Tabelle 2: Asymmetrische Verschlüsselung der XML-Elemente gemäß DeQS-RL

Bereich	Verfahren	Datenart	XML-Elemente	Öffentlicher Schlüssel der
Kranken-haus	direkt	QS-Daten	`qs_data`	BAS
	indirekt	QS-Daten	`qs_data`	DAS (LKG)
	Follow-up	QS-Daten	`qs_data`	DAS (LKG)
		PID-Daten	`patient`	VST-PSN
		PID TX-Register	`patient_tx`	VST-TX
kollektiv-vertraglich	Follow-up	QS-Daten	`qs_data`	DAS (KV)
		PID-Daten	`patient`	VST-PSN
selektivver-traglich	Follow-up	QS-Daten	`qs_data`	BAS
		PID-Daten	`patient`	VST-PSN

Tabelle 3: Asymmetrische Verschlüsselung der XML-Elemente gemäß oKFE-RL

Bereich	Datenart	XML-Elemente	Öffentlicher Schlüssel der
kollektivvertraglich	PB-Daten	`qs_data`	BAS
	PID-Daten	`patient`	VST

Beispiel 1: Anwender im stationären Sektor
Der Anwender ist im stationären Sektor tätig. Er muss die Elemente qs_data und patient auswählen und diese mit dem öffentlichen Schlüssel der Datenannahmestelle (qs_data) und der Vertrauensstelle (patient) verschlüsseln. Falls eine Datenlieferung für das Verfahren QS TX für den Versand vorbereitet werden soll, ist zwingend auch die Verschlüsselung des Elementes patient_tx erforderlich.

Beispiel 2: Anwender im ambulanten Sektor (kollektivvertraglich), Verfahren gem. DeQS-RL
Der Anwender ist im ambulanten Sektor tätig. Er muss die Elemente qs_data und patient auswählen und diese mit den öffentlichen Schlüsseln der zuständigen Datenannahmestelle (qs_data) und der Vertrauensstelle (patient) verschlüsseln.

Beispiel 3: Anwender im ambulanten Sektor (selektivvertraglich), Verfahren gem. oKFE-RL
Der Anwender ist im ambulanten Sektor tätig. Er muss die Elemente qs_data und patient auswählen und diese mit den öffentlichen Schlüsseln der Bundesauswertungsstelle (qs_data) und der Vertrauensstelle (patient) verschlüsseln.

Beispiel 4: Anwender im ambulanten Sektor, Verfahren gem. oKFE-RL
Der Anwender ist im ambulanten Sektor tätig. Er muss die Elemente qs_data und patient auswählen und diese mit den öffentlichen Schlüsseln der Bundesauswertungsstelle (qs_data) und der Vertrauensstelle (patient) verschlüsseln.

Beispiel 5: Versendung QS-Sollstatistik
Die QS-Sollstatistik wurde mit der Dokumentationssoftware erstellt und soll nun an die Datenannahmestelle verschickt werden.

Beispiel 6: Entschlüsselung Empfangsbestätigung/Datenflussprotokoll
Die Empfangsbestätigung oder das Datenflussprotokoll wurde zur Verfügung gestellt und muss nun entpackt werden.

9.2 Datenannahmestellen

Beispiel 7: Lieferung im stationären Sektor bearbeiten
QS-Daten müssen entschlüsselt werden

Beispiel 8: Lieferung im ambulanten Sektor (kollektivvertraglich), Verfahren gem. DeQS-RL
QS-Daten müssen entschlüsselt werden.

Beispiel 9: Lieferung im ambulanten Sektor (selektivvertraglich), Verfahren gem. DeQS-RL
Keine Entschlüsselung von QS-Daten.

Beispiel 10: Lieferung im ambulanten Sektor (kollektivvertraglich), Verfahren gem. oKFE-RL
Keine Entschlüsselung von PB-Daten.

Beispiel 11: Weiterleitung an VST (stationär)
QS-Daten aus dem stationären Sektor sowie pseudonymisierte LE-Kennungen zur Weiterleitung an die Vertrauensstelle verschlüsseln und verpacken.

Beispiel 12: Weiterleitung an VST (ambulant, kollektivvertraglich), Verfahren gem. DeQS-RL
QS-Daten aus dem ambulanten Sektor sowie pseudonymisierte LE-Kennungen zur Weiterleitung an die Vertrauensstelle verschlüsseln und verpacken.

Beispiel 13: Weiterleitung an VST (ambulant, selektivvertraglich), Verfahren gem. DeQS-RL
Pseudonymisierte LE-Kennungen zur Weiterleitung an die Vertrauensstelle verschlüsseln und verpacken.

Beispiel 14: Weiterleitung an VST (ambulant, kollektivvertraglich), Verfahren gem. oKFE-RL
Pseudonymisierte LE-Kennungen zur Weiterleitung an die Vertrauensstelle verschlüsseln und verpacken.

Beispiel 15: Lieferung Sollstatistik
Entschlüsselung der Sollstatistik.

Beispiel 16: Weiterleitung Sollstatistik an das IQTIG
Verschlüsselung der Sollstatistik zur Weiterleitung an das IQTIG.

Beispiel 17: Datenflussprotokoll des IQTIG entschlüsseln
Beispiel 18: Datenflussprotokoll des IQTIG für den LE verschlüsseln
Beispiel 19: Empfangsbestätigung der VST entschlüsseln