Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).



FAKULTÄT FÜR !INFORMATIK

Konzept für den kontrollierten Datentausch von (Energie-)Messdaten zwischen Kunden und Energieberater

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Werner Luckner

Matrikelnummer 0525048

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao. Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Mitwirkung: Projektass. Dipl.-Ing. Dr.techn. Alois Zoitl,
Dipl.-Ing. Dr. Johannes Goldynia (PDTS GmbH)

Wien, 16.3.2010

(Unterschrift Verfasser) (Unterschrift Betreuer)

Konzept für den kontrollierten Datentausch von (Energie-)Messdaten zwischen Kunden und Energieberater

Werner Luckner

16.3.2010

Kurzfassung

Um die Energiekosten eines Gebäudes nachhaltig und effektiv zu senken, ist eine qualitativ hochwertige Energieberatung notwendig. Damit hierbei das bestmögliche Ergebnis erreicht werden kann, sind möglichst genaue Verbrauchsmessdaten erforderlich. Dies ist allerdings mit den aktuellen Ableseintervallen (meist einmal jährlich) nicht zu erreichen. Daher beschäftigt sich die vorliegende Diplomarbeit mit der automatisierten Datenerfassung und Datenhaltung von Energieverbrauchsdaten (wie z.B. Strom-, Gas-, Wasser-, Fernwärmezählerständen). Dadurch kann erstens sehr viel häufiger der aktuelle Verbrauchswert ermittelt werden und zweitens muss der Energielieferant kein Personal mehr beschäftigen, welches in jedem Haushalt die Zähler händisch abliest.

Um unerlaubten Zugriff auf die Daten zu verhindern, findet die Speicherung der Daten verschlüsselt statt. Dabei wird ein hybrides Verschlüsselungsverfahren eingesetzt, welches eine Kombination aus einem symmetrischen und einem asymmetrischen Verfahren darstellt. Weiters wurde die Möglichkeit geschaffen, durch Gewährung des Zugriffs auf die entsprechenden Datensatzschlüssel des Kunden, einem bestimmten Energieberater den Zugriff auf ausgewählte Datensätze zu gewähren. Diese Daten können nun vom Berater dazu genutzt werden, ein Konzept für die Einsparung des Energieverbrauchs auszuarbeiten. Damit das bestmöglich durchgeführt werden kann, hat der Kunde die Möglichkeit, eine Beschreibung seiner Immobilien zu speichern und Dokumente zu diesen Immobilien hochzuladen. Auch diese Informationen kann der Kunde, durch Freigabe der entsprechenden Schlüssel, dem Berater zukommen lassen.

Diese Arbeit stellt ein Softwarekonzept für das eben beschriebene System dar. Dazu wird zuerst das gegebene Problem analysiert und allgemeine Designkonzepte vorgestellt. Danach werden die Abläufe der einzelnen Systemprozesse detailliert beschrieben. Weiters wird die Demo-Applikation vorgestellt, welche alle Funktionalitäten der Interfaces zwischen den Komponenten und alle Systemprozesse implementiert. Zugleich dient diese als Beweis für die Korrektheit des vorliegenden Konzepts. Anschließend wird die beschriebene Lösung aus verschiedenen Blickwinkeln diskutiert. Abschließend wird ein Vergleich des vorliegenden Systems mit dem *Google Powermeter* (Konkurrenzsystem) angestellt.

Die vorliegende Diplomarbeit bietet ein vollständiges Konzept zur Errichtung des beschriebenen Energieberatungsportals. Die reale Umsetzung des Konzepts würde die Qualität der Energieberatung erheblich verbessern und die Weitergabe der Daten vereinfachen. Die Verwendung automatischer Zähler bietet dem Energiekunden außerdem eine sehr viel bessere und genauere Übersicht über die verbrauchte Energie.

Abstract

To reduce the energy costs of a building sustainably and effectively a high quality energy consulting is necessary. To do this in the best way it is essential to get the best possible quality of data about the energy consumption. The actual intervals of reading the energy meters (typically once a year) are not sufficient to reach this quality. Because of that this master's thesis deals with the topic of automatic acquisition and storage of metered energy data (e.g. power-, gas-, water-, teleheating consumption). On the one hand this enables the customer to get meter data more often. On the other hand energy providers can save costs because they don't need staff for reading all the meters manually.

To avoid missuse of data, the data is stored in an encrypted way. For that encryption a hybrid encryption algorithm is used. This is a combination of a symmetric and an asymmetric algorithm. By granting the access to well-chosen private-keys of the datasets of the customer to an energy consultant, this consultant has the possibility to access the data of the customer. This data can now be used by the consultant to develop an energy concept for optimizing the energy consumption of the building. To do this in the best possible way, the customer has the possibility to save a precise description of the building. Moreover he can upload documents to his buildings. This datasets can also be shared to the consultant, by granting the access to the corresponding private keys.

This work represents a software concept for the system described above. At first an analysis of the problem and general design concepts are presented. After that the procedures of the particular system processes are described in detail. Hereafter the demo application is introduced, which implements all the functions of the interfaces and realizes all the procedures of the system processes. This demo application serves also a proof for the correctness of the concept. After describing the demo the given solution will be discussed. At the end a comparison with *Google Powermeter* (competitive system) is given.

The present master's thesis gives a full concept for the realizing of the described energy consulting portal. The implementation of this concept would significantly improve the quality of energy consulting and would simplify the process of sharing energy data. Additionally the energy customer will get a more exact overview about the energy consumption by the use of smart meters.

Inhaltsverzeichnis

Inhaltsverzeichnis **iii**

1 Einleitung **1**
1.1 Aufgabenstellung 2
1.2 Aufbau der Arbeit 2

2 Stand der Technik **3**
2.1 *Smart Metering* Systeme 3
2.2 Kryptographische Verfahren 4
2.2.1 Technologien zur Datenverschlüsselung 5
2.2.2 Digitale Signaturen 7
2.3 Web Services 9
2.3.1 SOAP 9
2.4 Zusammenfassung 11

3 Problemanalyse **13**
3.1 Ideen zur Lösung des Problems 13
3.2 Abläufe aus der Sicht des Kunden und des Energieberaters 15
3.2.1 Ablauf aus der Sicht des Kunden 15
3.2.2 Ablauf aus Sicht des Energieberaters 16
3.2.3 Use Cases 16
3.3 Verschlüsselung und Weitergabe der Daten eines Kunden an den Energieberater 19
3.4 Komponenten des Systems 20
3.5 Anforderungen an das Zielsystem 21
3.6 Zusammenfassung 22

4 Designkonzept **23**
4.1 Automatische Messwerterfassung 23
4.2 Daten eines Kunden 24
4.3 Verschlüsselung der Datensätze 26
4.4 Schlüssel eines Kunden 28
4.4.1 Backup-Schlüsselpaar 28
4.4.2 Schlüsselband (Wochenschlüssel) 29
4.4.3 Individuelles Datensatzschlüsselpaar 29

4.4.4 Metadaten-Schlüsselpaar 30
4.4.5 Signaturschlüsselpaar der Datenquelle 30
4.5 Speicherung der Metadaten 31
4.6 Der Lock-Mechanismus 34
4.6.1 Anforderung des Locks 34
4.7 Komponenten des Systems und deren Interaktion 36
4.7.1 Automatischer Zähler (Datenquelle) 36
4.7.2 Schlüsselverwaltung 37
4.7.3 Datenspeicher 37
4.7.4 Web-Portal (inkl. Clientkomponenten) 38
4.8 Kontaktaufnahme zwischen Kunden und Energieberatern 39
4.9 Weitergabe der Daten eines Kunden 39
4.9.1 Freigabestufen 40
4.10 Beschreibung der Systemprozesse 41
4.10.1 Use Case Diagramm 41
4.10.2 Systemprozess zur automatischen Messwerterfassung 43
4.11 Beschreibung der Interfaces 46
4.11.1 Versionierung der Schnittstelle 47
4.12 Historisierung des Datenspeichers 47
4.12.1 Hinzufügen eines Datensatzes 49
4.12.2 Update eines Datensatzes 49
4.12.3 Löschen eines Datensatzes 49
4.12.4 Lesen eines Datensatzes 49
4.12.5 Herstellen des Datenbestandes eines Kunden aus der Vergangenheit 49
4.12.6 Aufräumen der Historie 50
4.13 Zusammenfassung 50

5 Demo-Applikation **53**
5.1 Komponente *Schlüsselverwaltung* 54
5.1.1 Speicherung der Daten der Schlüsselverwaltung 56
5.2 Komponente *Datenspeicher 1* und Komponente *Datenspeicher 2* . . . 58
5.2.1 Speicherung der Daten in der Datenbank 59
5.3 Komponente *Kundenclient* 60
5.4 Komponente *Beraterclient* 60
5.5 Komponente *Automatischer Zähler* 61
5.6 Demonstration der Demo-Applikation anhand eines Sytemprozesses . 62
5.7 Zusammenfassung 70

6 Diskussion **71**
6.1 Konsistenzhaltung der Daten im Datenspeicher 71
6.2 Verschlüsselte Metadaten vs. Unverschlüsselte Metadaten 73
6.3 Verbesserung des Lock-Mechanismus 76
6.4 Speicherbedarf 78
6.4.1 Speicherbedarf im Datenspeicher bei nicht historisierter Speicherung 79

6.4.2 Speicherbedarf im Datenspeicher bei historisierter Speicherung 83
6.4.3 Speicherbedarf in der Schlüsselverwaltung 89
6.5 Zusammenfassung 89

7 Vergleich mit dem Google PowerMeter 91
7.1 Vergleich: Google PowerMeter vs Open Facility Data Portal 93

8 Zusammenfassung und Ausblick 95

A Anhang 99
A.1 Stichwortverzeichnis 99
A.2 Abkürzungsverzeichnis 102
A.3 Listing der XML-Schemas des Datensätze eines Kunden 103
A.3.1 Kundeninformationsdatensatz 103
A.3.2 Immobiliendatensatz 103
A.3.3 Dokumentendatensatz 103
A.3.4 Zählerinformationsdatensatz 103
A.3.5 Parameterdatensatz 104
A.3.6 Messwertdatensatz 104
A.4 Listing des XML-Schemas der Metadaten-Datenstruktur 105
A.5 Beschreibung der Systemprozesse 106
A.5.1 Ablauf der Systemprozesse 106
A.5.2 Erklärung der verwendeten Symbole 106
A.5.3 Messwerterfassung durch eine automatische Datenquelle . . . 106
A.5.4 Dateneinbringung durch den Kunden 111
A.5.5 Datenabfrage 124
A.5.6 Datenanforderung und -freigabe 125
A.5.7 Editieren von Datenobjekten 135
A.5.8 Sonstige Systemprozesse 136
A.6 Interfacebeschreibung 143
A.6.1 Interface zwischen Client und Schlüsselverwaltung 143
A.6.2 Interface zwischen automatischer Datenquelle und Schlüsselverwaltung 160
A.6.3 Interface zwischen Datenspeicher und Schlüsselverwaltung . . 162
A.6.4 Interface zwischen Client und Datenspeicher 163
A.6.5 Interface zwischen automatischer Datenquelle und Datenspeicher 172
A.7 Listings der WSDL-Files und XML-Schemas der Interfaces 180
A.7.1 Interface zwischen Client und Schlüsselverwaltung 180
A.7.2 Interface zwischen automatischen Datenquellen und Schlüsselverwaltung 199
A.7.3 Interface zwischen Datenspeicher und Schlüsselverwaltung . . 203
A.7.4 Interface zwischen Client und Datenspeicher 207
A.7.5 Interface automatischen Datenquellen und Datenspeicher . . . 221
A.8 Inbetriebnahme der Demo-Applikation 235
A.9 Listings der Skripts zum Erstellen der Datenbanktabellen 237

A.9.1 Tabellen für die *Schlüsselverwaltung* 237
A.9.2 Tabellen für *Datenspeicher 1* 239
A.9.3 Tabellen für *Datenspeicher 2* 239

Literaturverzeichnis **241**

KAPITEL 1

Einleitung

Aufgrund der steigenden Energiekosten und des anwachsenden Energiebedarfs wird Energieeffizienz immer wichtiger. Um diese zu erreichen sind zum einen bauliche Maßnahmen an Gebäuden zu treffen und zum anderen auf möglichst energieeffiziente Verbrauchsgeräte zu achten. Auch ein gesteigertes Bewusstsein der Energiekunden ist notwendig, um nachhaltig den Energieverbrauch und die damit verbundenen Energiekosten zu senken.

Um all dies umzusetzen, ist eine qualitativ hochwertige Energie- und Bauberatung von Nöten. Diese ist allerdings nur durch hochwertige Verbrauchsdaten zu gewährleisten. Dazu sind die momentanen Ableseintervalle, welche primär zur Verrechnung herangezogen werden, keinesfalls ausreichend und daher für eine gute Energieberatung ungeeignet. Daher sollen in Zukunft automatische Verbrauchszähler (sogenannte *Smart Meter*) eingesetzt werden, welche die Verbrauchsdaten automatisch regelmäßig ablesen und in ein Datenerfassungssystem einbringen. Dies sollte im Idealfall alle Verbrauchszähler einer Immobilie bzw. Anlage umfassen. Dazu gehören neben dem Stromzähler auch Gas-, Wasser- und Fernwärmezähler.

Die Idee der automatischen Zähler ist nicht neu. Allerdings handelt es sich bei den momentan vorhanden Lösungen meist nur um automatische Stromzähler. Außerdem handelt es sich meist um proprietäre Systeme eines Anbieters. Dies führt bei einem Wechsel des Energieanbieters daher auch meist zu einem Wechsel des Systems [10].

Da es sich bei diesen Daten um sensible Daten handelt, müssen diese mit höchster Vorsicht aufbewahrt werden. Mit diesen Daten könnte beispielsweise unerlaubt das Nutzerverhalten eines Energiekunden ausgeforscht werden. Damit dies ausgeschlossen werden kann, sollen die Daten eines Kunden verschlüsselt in den Datenspeichern abgelegt werden. Dies bedeutet, dass unberechtigte Personen und selbst die Anbieter der Datenspeicher die Daten nicht auswerten können. Somit wäre ein Missbrauch der Daten ausgeschlossen.

Da für die effiziente Energieberatung diese Daten notwendig sind, muss eine Möglichkeit geschaffen werden, diese Daten an den Berater weiterzugeben. Allerdings

muss natürlich darauf geachtet werden, dass nur dieser ausgewählte Berater Zugriff auf die Daten hat. Alle anderen Personen sollen weiterhin keinen Zugriff haben.

1.1 Aufgabenstellung

Aufgabe dieser Arbeit ist die Erstellung eines vertikal durchgängigen Designkonzepts eines offenen Datenerfassungs- und Beratungsportals im Sinne des *„White Book - Open Facility Data Platform“* [10]. Dabei soll die Energieverbrauchsdatenerfassung sowohl durch automatische Zähleinrichtungen, als auch durch manuelle Eingabe durch den Kunden stattfinden können. Wesentlicher Bestandteil soll ein kryptografisches Konzept sein, welches es ermöglicht, alle Daten eines Kunden verschlüsselt im System zu speichern. Außerdem soll eine Möglichkeit geschaffen werden, einem ausgewählten Energieberater bestimmte Daten zur Verfügung zu stellen (siehe hierzu Abbildung 1.1).

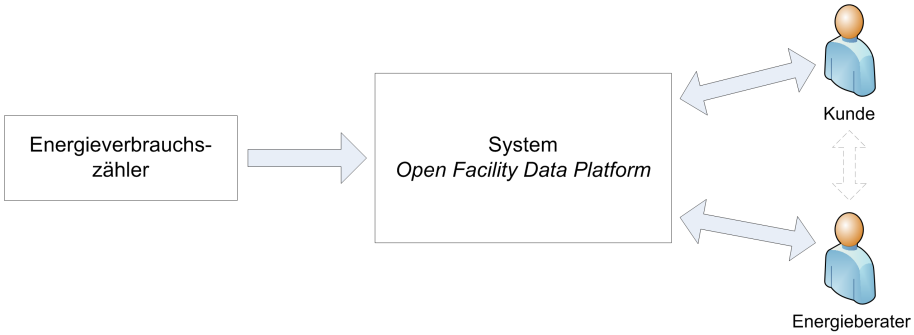


Abbildung 1.1: Aufgabenstellung

1.2 Aufbau der Arbeit

In Abschnitt 2 wird ein Überblick über Themenbereiche *Smart Metering*, *Verschlüsselung von Daten* und *Web Services* gegeben. Anschließend wird in Abschnitt 3 das vorliegende Problem analysiert und erste Ideen zur Lösung vorstellt. Danach wird in Kapitel 4 das Designkonzept genau beschrieben. Hiernach wird in Kapitel 5 eine Demo-Applikation vorgestellt, welche die Funktionsfähigkeit des Designs bestätigt. Anschließend wird in Abschnitt 6 die vorgestellte Lösung eingehend diskutiert. Danach wird das vorliegende System mit dem *Google Powermeter* verglichen, welches ähnliche Ziele wie das vorliegende System verfolgt. Abschließend werden die Ergebnisse zusammengefasst und Anregungen für zukünftige Weiterentwicklungen des Systems aufgezeigt.

Hinweis: Alle personenbezogenen Begriffe in dieser Arbeit sind geschlechtsneutral zu verstehen.

KAPITEL 2

Stand der Technik

In diesem Abschnitt soll ein kurzer Überblick über bestehende Systeme im Bereich des *Smart Metering*s gegeben werden. Da hierbei der Datenschutz ein wichtiges Thema ist und daher auch die Verschlüsselung und Signatur von Daten einen Kernpunkt dieser Arbeit darstellt, wird außerdem ein Überblick über die momentan üblichen Datenverschlüsselungsverfahren gegeben. Da in dieser Arbeit auch Web Services eine große Rolle spielen, wird dieses Thema ebenfalls behandelt.

2.1 *Smart Metering* Systeme

In den letzten Jahren wird das Bestreben, die alten mechanischen Stromzähler (auch *Ferrariszähler* genannt) durch neue intelligente Zähler (*Smart Meters*) auszutauschen, immer größer. Die Gründe hierfür sind vielfältig: Ein Grund ist, dass durch den Einsatz von *Smart Meters* das jährliche Ablesen des Zählerstandes für die Verrechung nicht mehr händisch augeführt werden muss, sondern dass der Zähler seine Verbrauchswerte automatisch an den Energieversorger sendet. Ein weiterer wichtiger Grund ist, dass sich die Energieversorger erhoffen, dass durch den Einsatz von *Smart Meters* der gesamte Stromverbrauch in Zaum gehalten werden kann. So erhofft man sich in Privathaushalten, dass der Energiebedarf um $5\% - 10\%$ gesenkt werden kann [22]. Ein Grund für diese Einsparung ist die Bewusstmachung der Verbrauchswerte beim Kunden. So kann sich der Kunde jederzeit über ein Webportal oder eine andere Anzeige im Haushalt seine aktuellen Verbrauchsdaten und deren Historie anzeigen lassen.

Darüber hinaus erhoffen sich die Energieversorger eine bessere Lastverteilung, sodass energieintensive Verbraucher erst dann betrieben werden, wenn das Netz nur wenig belastet ist. Zu diesen Verbrauchern zählen vor allem Waschmaschinen, Wäschetrockner und Geschirrschülmaschinen. Damit der Kunde daraus einen Nutzen erfährt, wird es zukünftig variable Stromtarife geben, welche sich mit der Last des Netzes automatisch anpassen. So wird Strom zu Zeiten, in welchen das Netz Spitzenlasten zu bewältigen hat, teurer sein als zu Zeiten, in welchen eine normale oder nur geringe

Belastung herrscht. Dies bedeutet, dass der Zähler die Möglichkeit bieten muss, den aktuellen Stromtarif zu erfahren und darüber hinaus bestimmte Verbraucher erst dann einzuschalten, wenn ein günstiger Tarif angeboten wird. Dies bedarf natürlich auch einer Anpassung dieser Verbrauchsgeräte.

Nicht zuletzt sei erwähnt, dass das *3. EU-Binnenmarktpaket* in *Anhang A* (siehe [1]) die Einführung von intelligenten Zählern vorschreibt. So sollen bis 2020 80% der Stromkunden mit einem Smart Meter ausgestattet sein (siehe hierzu [9]). So wurden in der letzten Zeit von einigen Energieversorgern in Österreich Feldversuche gestartet, in welchen ein Teil der versorgten Haushalte mit *Smart Meters* gestattet wurden. Dabei kommen mehrere verschiedene Systeme zum Einsatz. Aufgrund der soeben genannten EU-Richtlinie ist man allerdings nun bestrebt, die Infrastruktur hinsichtlich des Einsatzes von intelligenten Stromzählern zu vereinheitlichen und somit einen gemeinsamen Standard zu schaffen, welcher dann österreichweit eingehalten wird.

Smart Metering ist allerdings nicht auf Stromzähler allein begrenzt. Zwar waren diese die ersten Zähler, welche automatisch abgelesen wurden, aber die Idee lässt sich auf alle anderen Verbrauchszähler im Haushalt ausweiten. So ist von einigen Energieversorgern zu hören, dass das System auf Gas- und Wasserzähler ausgeweitet werden soll [19].

Zur Einführung von *Smart Meters* gibt es allerdings auch einige kritische Stimmen. So lässt sich beispielsweise durch die relativ kleinen Ableseintervalle ein Verhaltensprofil des Kunden erstellen, wodurch sich Rückschlüsse auf die Lebensgewohnheiten der Kunden gewinnen lassen. Solche Datensammlungen sind Datenschützern natürlich ein Dorn im Auge. Deshalb ist beim Design der Infrastruktur für intelligente Zähler besonders zu beachten, dass unbefugte Dritte niemals Zugriff auf diese Daten erhalten. Wie eine Studie des amerikanischen Sicherheitsunternehmens IOActive zeigt, ist dies aber keinesfalls selbstverständlich (siehe Pressemeldung zur Studie [14]). Diese Studie besagt, dass der in den USA geplante *Smart Grid* erhebliche Sicherheitslücken aufweist. Somit könnten sich Hacker mit relativ geringen Mitteln Zugriff auf das System verschaffen und Daten auslesen und sogar so manipulieren, dass die Stromversorgung beeinträchtigt wird. Dies geschieht durch Signalisierung einer nicht vorhandenen Spitzenlast, welche zu einer Reaktion beim Energieversorger führt, welche eine Überspannung im Netz hervorruft (vgl. [11]).

Wie hier gezeigt wurde, stellt das *Smart Metering* einen spannenden Bereich dar, welcher sich momentan in einem starken Aufbruch befindet. Es bleibt also abzuwarten, welche Innovationen und Ideen sich in diesem Themenfeld zukünftig entwickeln.

2.2 Kryptographische Verfahren

In diesem Abschnitt sollen die Grundlagen kryptographischer Verfahren vorgestellt werden. Dazu werden zuerst die wichtigsten Technologien zur Datenverschlüsselung beschrieben. Anschließend wird auf das Thema digitale Signaturen näher eingegangen. Zusammen stellen diese beiden Techniken sicher, dass der Inhalt einer Nachricht abhörsicher (Verschlüsselung), unverändert und ungefälscht (digitale Signatur) beim Empfänger eintrifft.

2.2.1 Technologien zur Datenverschlüsselung

Unter Verschlüsselung von Daten versteht man eine mathematische Transformation von lesbaren Daten (Klartext) in einen Geheimtext. Diese Transformation soll dabei so gewählt werden, dass alleine aus dem Geheimtext der Klartext nicht wieder hergestellt werden kann. Nur wenn dem Empfänger der Geheimtexts zusätzliche Informationen bekannt sind, kann dieser eine Rücktransformation durchführen, sodass er aus dem Geheimtext wieder den Klartext herstellen kann. Diese zusätzlichen Informationen werden in Kryptosystemen in der Regel als Schlüssel bezeichnet.

Der Zweck von Verschlüsselungsverfahren ist jener, dass unbefugte Personen bestimmte Informationen nicht lesen können. Das heißt, dass ein Sender, welcher einem Empfänger eine vertrauliche Nachricht zukommen lassen möchte, diese Nachricht vor dem Versenden verschlüsselt. Danach kann diese verschlüsselte Nachricht dem Empfänger übermittelt werden. Dazu kann auch ein unsicheres Medium genutzt werden. Sollte diese Nachricht abgehört oder sonst auf irgendeine Weise abgefangen werden, so kann dieser Angreifer mit der Nachricht nichts anfangen, da er die Klartextinformation gar nicht oder zumindest nicht mit vertretbarem Aufwand wiederherstellen kann.

Prinzipiell kann zwischen zwei großen Gruppen von Verschlüsselungsverfahren unterschieden werden. Diese sind einerseits symmetrische Verfahren und andererseits asymmetrische Verfahren. Darüber hinaus gibt es noch hybride Verfahren, welche eine Kombination der beiden bilden. Im weiteren werden diese Arten der Verschlüsselungsverfahren näher erklärt und Unterschiede aufgezeigt.

Symmetrische Verschlüsselungsverfahren

Das wesentlichste Merkmal von symmetrischen Verschlüsselungsverfahren ist jenes, dass zur Verschlüsselung und zur Entschlüsselung der selbe Schlüssel verwendet wird. Dieser Schlüssel wird meist *geheimer Schlüssel* (oder englisch *Secret-Key*) genannt. Abbildung 2.1 stellt die Vorgänge grafisch dar (siehe [18]).

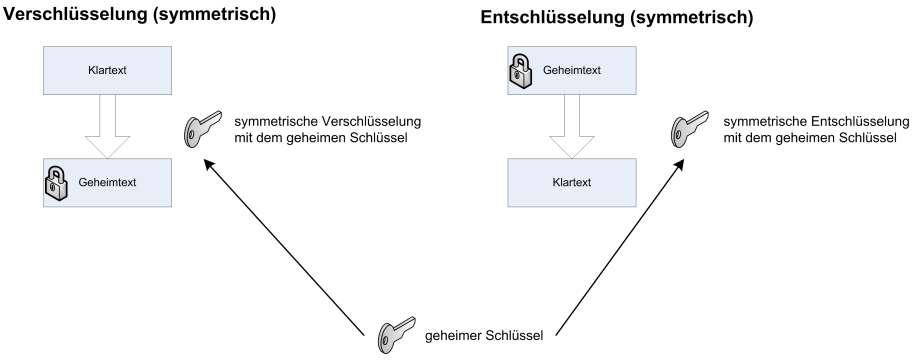


Abbildung 2.1: Symmetrische Verschlüsselung

Der große Vorteil von symmetrischen Kryptoverfahren ist, dass diese meist sehr effizient implementiert werden können und daher auch sehr schnell sind. So können diese Verfahren auch zur Verschlüsselung von Internetstreams in Echtzeit eingesetzt werden. Allerdings haben symmetrische Verfahren den Nachteil, dass der Schlüsseltausch nicht so einfach durchgeführt werden kann, wie bei asymmetrischer Verschlüsselung. Entweder der geheime Schlüssel wird vorher außerhalb des Kommunikationssystems ausgemacht oder es muss ein Schlüsseltauschprotokoll implementiert werden. Dieses soll es ermöglichen, einen geheimen Schlüssel zu vereinbaren, ohne dass dieser von unbefugten Dritten, welche eventuell die Leitung abhören, ermittelt werden kann.

Beispiele für moderne symmetrische Verschlüsselungsalgorithmen sind DES[1], Triple-DES (auch 3DES), AES[2] oder IDEA[3] [18].

Asymmetrische Verschlüsselungsverfahren

Bei asymmetrischen Verschlüsselungsverfahren (auch Public-Key-Verfahren genannt) werden zur Verschlüsselung und zur Entschlüsselung unterschiedliche Schlüssel eingesetzt. Diese Schlüssel bilden ein Schlüsselpaar, wobei der Schlüssel zur Verschlüsselung *Public-Key* genannt wird. Der Schlüssel zur Entschlüsselung wird als Private-Key bezeichnet. Bei der Genierierung des Schlüssels muss darauf geachtet werden, dass es unmöglich ist, aus dem Public-Key den Private-Key abzuleiten [18]. Abbildung 2.2 stellt die asymmetrische Verschlüsselung grafisch dar.

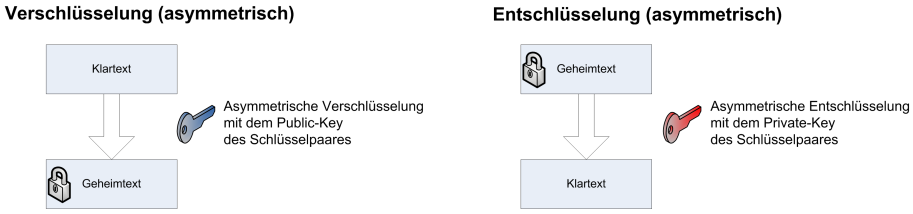


Abbildung 2.2: Asymmetrische Verschlüsselung

Der wesentlichste Vorteil der asymmetrischen Verschlüsselung ist, dass der Schlüsseltausch wesentlich einfacher realisiert werden kann. Der Besitzer des Schlüsselpaares (Empfänger der Nachricht) kann dabei den Public-Key jedem frei zur Verfügung stellen. Der Private-Key hingegen darf niemals frei zugänglich gemacht werden. Möchte nun jemand eine vertrauliche Nachricht an den Empfänger senden, so kann er diese mit dem frei verfügbaren Public-Key verschlüsseln und anschließend übertragen. Der Empfänger kann nach dem Empfang der Nachricht diese mit seinem Private-Key entschlüsseln und somit wieder im Klartext lesen. Der große Nachteil von asymmetrischen Verfahren ist, dass sie meist nicht so effizient umsetzbar sind wie symmetrische Verfahren.

[1] Data Encryption Standard
[2] Advanced Encryption Standard
[3] International Data Encryption Algorithm

Das wohl bekannteste *asymmetrische* Verfahren ist das RSA[4]-Verfahren [4].

Hybride Verschlüsselungsverfahren

Hybride Verfahren bilden eine Kombination zwischen symmetrischen und asymmetrischen Verfahren. Sie vereinen die Vorteile der beiden Verfahren. Dazu wird die eigentliche Nachricht mit einem symmetrischen Schlüssel verschlüsselt, welcher meist erst bei der Verschlüsselung erzeugt wird. Dieser Schlüssel wird meist *Session-Key* genannt. Anschließend wird der Session-Key mit dem Public-Key des Emfpängers verschlüsselt und der verschlüsselte Session-Key an die Nachricht angehängt, welche später übertragen werden kann. Beim Empfänger wird schließlich der verschlüsselte Session-Key mit dem Private-Key des Empfängers entschlüsselt. Anschließend kann mit dem Session-Key die eigentliche Nachricht wieder entschlüsselt werden. Abbildung 2.3 zeigt den Ablauf bei hybriden Verschlüsselungsverfahren (siehe [18], [4]).

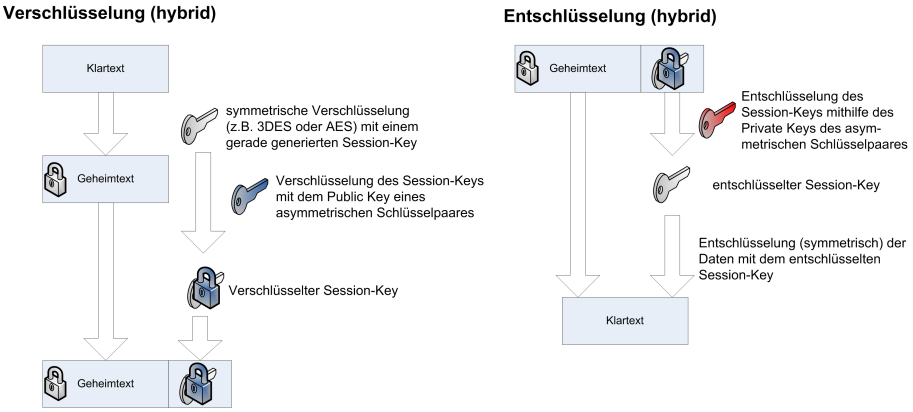


Abbildung 2.3: Hybrides Verschlüsselungsverfahren

Dieses Verfahren kann nun ähnlich effizient implementiert werden, wie ein symmetrisches Verfahren, da nur sehr kleine Datenmengen (der Session-Key) asymmetrisch verschlüsselt werden müssen. Der Schlüsseltausch kann genauso wie bei asymmetrischen Verfahren durchgeführt werden. Als einzige Nachteil kann hier erwähnt werden, dass das Verfahren etwas komplizierter ist, da sowohl ein symmetrisches als auch ein asymmetrisches Verfahren verwendet wird.

2.2.2 Digitale Signaturen

Verschlüsselung alleine reicht für eine sichere Datenübertragung nicht aus. So könnte bei asymmetrischer oder hybrider Verschlüsselung beispielsweise der Absender der Nachricht gefälscht werden. Auch könnte es vorkommen, dass ein Angreifer die Nachricht abfängt und den Inhalt der Nachricht selbst austauscht. Dazu muss der Angreifer lediglich die falsche Nachricht mit dem Public-Key des Empfängers verschlüsseln

[4]benannt nach den Erfindern Ronald L. Rivest, Adi Shamir und Leonard Adleman

und diese statt der originalen Nachricht weiter an den Empfänger senden. Ein solcher Angriff wird *Man in the middle*-Angriff genannt.

Um einen solchen Angriff zu verhindern, wurden digitale Signaturen entwickelt. Um eine Nachricht zu signieren, wird als erstes mithilfe einer *Einweg-Hash-Funktion* ein Hashwert gebildet. Eine solche Hash-Funktion ist eine mathematische Abbildung, welche folgende Eigenschaften erfüllt [13]:

- Kompression: unabhängig von der Größe der Nachricht, besitzt der Hashwert eine bestimmte Länge
- Unumkehrbarkeit: vom Hashwert kann nicht auf die Nachricht geschlossen werden
- Kollisionsvermeidung: es soll sichergestellt werden, dass keine zwei Nachrichten gefunden werden können, welche den gleichen Hashwert ergeben

Die bekanntesten Hash-Funktionen heißen MD5[5] und SHA[6]. Nachdem der Hashwert berechnet wurde, verschlüsselt der Sender diesen mit einem asymmetrischen Verschlüsselungsverfahren mithilfe seines eigenen Private-Keys. Dieser verschlüsselte Hashwert bildet nun die Signatur der Nachricht. Nun wird die Nachricht zusammen mit der Signatur an den Empfänger versendet. Der Empfänger berechnet aus der Nachricht mit der selben Einweg-Hashfunktion den Hashwert der empfangenen Nachricht. Weiters entschlüsselt der Empfänger die Signatur mit dem Public-Key des Senders. Abschließend vergleicht er diese beiden Hashwerte [18]. Stimmen diese überein, so wurde die Nachricht korrekt empfangen. Somit kann sich der Empfänger sicher sein, dass die Nachricht unverfälscht angekommen ist (Integrität) und der Absender der Nachricht nicht gefälscht wurde. Andernfalls hat bei der Übertragung eine Manipulation stattgefunden.

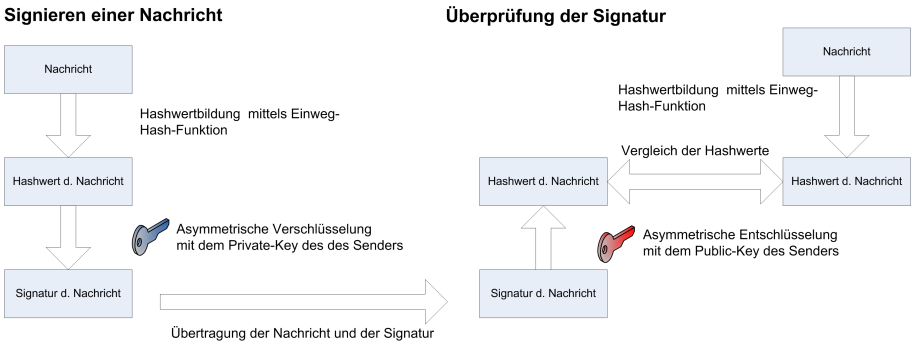


Abbildung 2.4: Digitale Signatur

[5] Message Digest 5
[6] Secure Hash Algorithm

2.3 Web Services

Auf den Seiten des *World Wide Web Consortiums* wird der Begriff des *Web Services* folgendermaßen definiert:

Zitat aus [21]: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Was sagt diese Definition nun alles aus? In dieser Definition steht als erstes, dass es sich bei Web Services um Software handelt, welche die Interaktion zwischen Maschinen über ein Netzwerk erlaubt. Ein wichtiger Punkt hierbei ist allerdings die Interoperabilität. Dies bedeutet, dass auch diese Interaktion zwischen verschiedenartigen Maschinen stattfinden kann. Diese müssen dabei nur einen gemeinsamen Standard einhalten. Damit ist gemeint, dass sowohl das Betriebssystem als auch die Programmiersprache des Clients und des Servers komplett verschieden sein können. So kann beispielsweise ein Windows-Client, welcher in C# programmiert wurde, problemlos mit einem Web Service interagieren, welches in Java programmiert ist und in einer Linux Umgebung läuft.

Weiters besagt die Definition, dass die Schnittstelle solcher Web Services in einer maschinenverständlichen Sprache in Form von WSDL[7]-Files vorliegt. Dies bedeutet, dass der Client des Services durch laden dieses WSDL-Files die komplette Schnittstelle kennenlernen und somit mit dem Server kommunizieren kann. Ein weiterer wichtiger Punkt ist die Kommunikation über SOAP[8]-Nachrichten, auf welche später noch kurz eingegangen wird. Als Übertragungsprotokoll wird meist das HTTP[9]-Protokoll eingesetzt, wobei die Nachrichten ins XML[10]-Format serialisiert werden. Der Vorteil der Verwendung von HTTP als Übertragungsprotokoll ist jener, dass der Port für HTTP Übertragungen bei den meisten Firewalls nicht blockiert wird und es somit bei der Übertragung kaum zu Schwierigkeiten kommt [12].

Darüber hinaus können noch einige andere Web-Standards zum Einsatz kommen. Mit diesen Standards sind beispielsweise alle WS-*-Standards gemeint. Diese beinhalten beispielsweise Standards für die Verschlüsselung und Signatur von Nachrichten oder Standards zur Adressierung oder Transaktionalität.

2.3.1 SOAP

In diesem Abschnitt soll kurz erklärt werden was man unter dem Begriff SOAP versteht und wie SOAP funktioniert.

[7] Web Service Description Language
[8] Simple Object Access Protocol
[9] Hypertext Transfer Protocol
[10] Extensible Markup Language

SOAP ist ein Protokoll, welches XML-basierten Nachrichtenaustausch reglementiert. SOAP stellt dabei einen W3C Standard dar. Die aktuelle Version ist die Version 1.2 [24]. Wie zuvor schon erwähnt, werden SOAP-Nachrichten meist über HTTP übertragen. Es sind allerdings auch andere Übertragungsprotokolle, wie SMTP/POP3, FTP oder JMS möglich. SOAP ermöglicht es, auf einer entfernten Maschine eine Funktion aufzurufen (*Remote Procedure Calls*). Dabei reglementiert SOAP die Syntax der Anfragen und Ergebnisse dieser Funktionsaufrufe [12]. Die größten Vorteile von SOAP sind Plattform- und Programmiersprachenunabhängigkeit, Offenheit, Robustheit und Skalierbarkeit. Als größter Nachteil von SOAP ist der große Overhead zu erwähnen. So kann es vorkommen, dass eine Nachricht, welche nur einen Byte-Wert beinhaltet, einige hundert Byte groß ist. Ein weiterer Nachteil ist der damit verbundene Rechenaufwand beim Generieren und Parsen dieser großen XML-Nachrichten.

Aufbau von SOAP-Nachrichten

Jede SOAP Nachricht besteht aus einem sogenannten *Envelope*, welches das XML-Dokument als SOAP-Nachricht identifiziert [25]. Dieses Envelope beinhaltet ein Header-Element und ein Body-Element, wobei das Header-Element optional ist. Im Header können Kontrollinformationen zur Weiterleitung und Bearbeitung der Nachricht stehen [24]. Der Body enthält die eigentliche Nachricht, also beispielsweise einen Funktionsaufruf oder dessen Ergebnis. Darüber hinaus kann der Body noch ein Fault-Element beinhalten, welches die Gegenstelle über einen Fehler informiert. Das anschließende Listing beinhaltet ein Beispiel eines Funktionsaufrufs mittels SOAP und dessen Antwort. Dieses Beispiel wurde von [25] übernommen. In der Anfrage wird der Preis für Äpfel angefragt. Die Antwort liefert schließlich diesen Preis zurück.

Anfrage:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Antwort:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

2.4 Zusammenfassung

In diesem Abschnitt wurde ein Überblick über die Themenbereiche *Smart Metering*, Technologien zur Datenverschlüsselung und *Web Services* gegeben. Smart Metering ist ein Thema, welches in den letzten Jahren immer mehr an Relevanz gewonnen hat. So sollen, nicht zuletzt wegen der angesprochenen EU-Richtlinie, in naher Zukunft die alten Verbrauchszähler gegen neue moderne Zähler ausgetauscht werden, welche es ermöglichen, den Zählerstand und dessen Historie über das Internet abzufragen. Dabei sollen diese auch Smart Grid Funktionalitäten erhalten. Dazu gehört beispielsweise, dass gewisse Verbrauchsgeräte erst dann eingeschaltet werden, wenn das Netz wenig belastet ist und somit auch der Tarif niedriger ist. Somit lassen sich für den Verbraucher auch Kostenvorteile erzielen.

Bei den Technologien zur Datenverschlüsselung wurde zuerst die symmetrische Verschlüsselung vorgestellt, welche zur Ver- und Entschlüsselung der Daten den gleichen geheimen Schlüssel verwendet. Anschließend wurde das asymmetrische Verschlüsselungsverfahren vorgestellt. Dieses nutzt zur Ver- und Entschlüsselung unterschiedliche Schlüssel, welche zusammen ein Schlüsselpaar bilden. Die asymmetrische Verschlüsselung ist gegenüber der symmetrischen Verschlüsselung zwar langsamer, hat aber den Vorteil des einfacheren Schlüsseltauschs. Danach wurde noch das hybride Verschlüsselungssystem vorgestellt, welches eine Kombination von symmetrischer und asymmetrischer Verschlüsselung darstellt. Dieses System vereint dabei die Vorteile dieser beiden Verfahren. Nach den Technologien zur Datenverschlüsselung wurde auf digitale Signaturen eingegangen. Diese Stellen sicher, dass eine Nachricht unverfälscht übertragen wurde. Außerdem kann durch die Signatur die Herkunft der Nachricht sichergestellt werden.

Abschließend wurde der Begriff eines Web Services erklärt. Dabei wurde vor allem auf die Übertragung der Daten und auf das Nachrichtenprotokoll SOAP eingegangen.

KAPITEL 3

Problemanalyse

In diesem Kapitel soll vermittelt werden, wie an die gestellte Aufgabe herangegangen wurde. Dazu wurde zuerst eine *Mind Map* erstellt, welche die Basis für alle nachfolgenden Schritte bildet. Danach wurde versucht, aus dieser Mind Map erste grobe und noch sehr abstrakte Abläufe und Use Cases aus der Sicht der Energiekunden und aus der Sicht der Energieberater herauszufiltern. Danach werden noch einige Ideen zur Verschlüsselung und Weitergabe der Daten eines Kunden an den Energieberater zusammengefasst. Anschließend werden die Kernkomponenten identifiziert und deren Aufgaben grob dargelegt. Als Ergebnis des Analyseprozesses wurde eine Liste von Anforderungen definiert, welche das Zielsystem später erfüllen soll.

3.1 Ideen zur Lösung des Problems

Als erster Schritt wurde eine sogenannte *Mind Map* (Abbildung 3.1) erstellt, welche alle Ideen zur Lösung der Aufgabe vereint. Dies ist noch eine sehr abstrakte und teilweise noch etwas unsortierte Form des Ideensammelns. Außerdem muss hierbei nicht jede Idee, welche in der Mind Map auftaucht, auch in der späteren Lösung umgesetzt werden. Es ist nicht einmal gesagt, dass diese Ideen überhaupt umsetzbar sind.

Kurz zusammengefasst wurden aus diesem Schritt folgende Ergebnisse gewonnen: Der Kunde kann Profile zu seiner Person und zu allen seinen Immobilien anlegen. Außerdem kann der Kunde ergänzende Dokumente hochladen. Der Kunde kann zu jeder seiner Immobilien Zähler anlegen, welche entweder automatisch Zählwerte in das System einbringen oder der Kunde trägt diese händisch ein, nachdem er den Zählerstand abgelesen hat. Alle Daten eines Kunden werden verschlüsselt gespeichert. Dabei könnten bei der Verschlüsselung der Messdaten wöchentlich wechselnde Schlüssel verwendet werden, welche zur Datenfreigabe an den Energieberater übergeben werden. Der Kunde kann individuell entscheiden, welche Datensätze er an welchen Berater weitergibt. Außerdem muss eine Komponente existieren, welche die Kommunikation zwischen Energiekunden und Beratern ermöglicht.

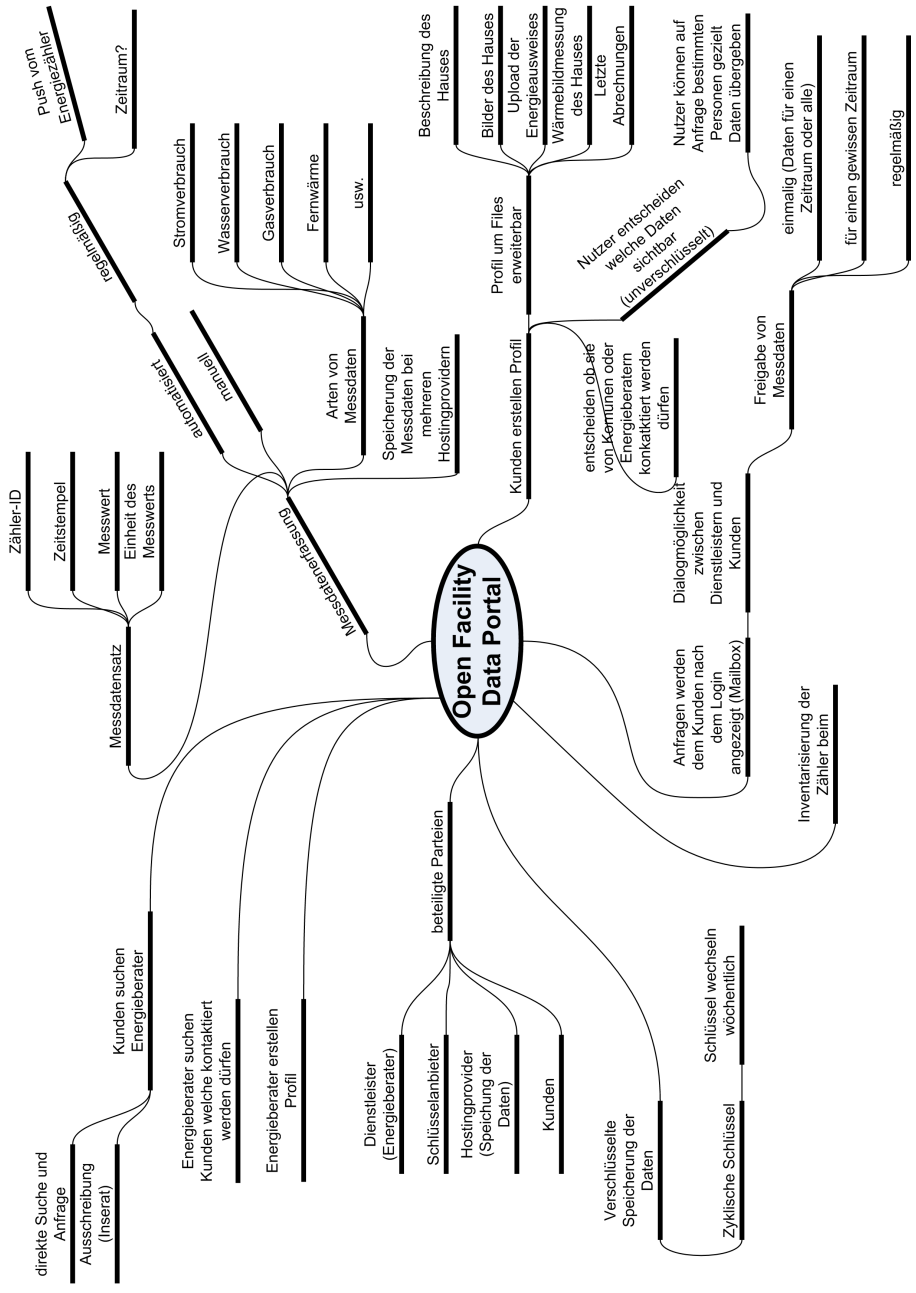


Abbildung 3.1: Mind Map

3.2 Abläufe aus der Sicht des Kunden und des Energieberaters

Nach der Erstellung der Mind Map wurde versucht, die daraus gewonnene Information in eine sortiertere Form zu fassen und diese eventuell schon etwas zu verfeinern. Dazu wurden grobe Abläufe aus der Sicht des Kunden und aus der Sicht des Energieberaters definiert. Dabei wird beschrieben, welche Aktionen und in welcher Reihenfolge durch den Kunden zu setzen sind, um Daten einem Energierater zukommen zu lassen, um anschließend von ihm beraten zu werden. Anschließend wird der Ablauf einer Beratung aus der Sicht des Energieberaters beschrieben.

Die Schritte der Abläufe aus der Sicht des Kunden und aus der Sicht des Energieberaters müssen nicht genau in dieser Reihenfolge ausgeführt werden. Es handelt sich hierbei um Beispielabläufe.

3.2.1 Ablauf aus der Sicht des Kunden

1. Kunde registriert sich und legt Immobilien an
   a) Kunde gibt seine Nutzerdaten ein
   b) Kunde legt Immobilien an (Häuser, Firmengebäude, usw.)
   c) Kunde gibt Daten zu den verschiedenen Immobilien ein (Art des Objekts, Anzahl der Personen im Haushalt, Größe, Bauweise, Art der Heizung,...)
   d) Kunde lädt zu den Immobilien weitere Dokumente wie Fotos, Energieausweis, Wärmebildmessungen, usw. hoch
   e) Kunde entscheidet, welche Daten für einen bestimmten oder alle Energieberater lesbar sind
   f) Kunde entscheidet, ob er von Energieberatern gefunden und kontaktiert werden darf
2. Zu jedem Objekt werden Zähler inventarisiert:
   a) Hostingprovider wird festgelegt
   b) Art des Zählers (Strom, Wasser, Gas,...)
   c) Messwerteinheit
   d) Zähler-ID
   e) Kommentar
3. Zähler werden angeschlossen und Messwerte in der Datenbank des Datenspeichers gespeichert
4. Der Kunde kann seine aktuellen Zählerstände und Verbrauchswerte tabellarisch abfragen.

   Außerdem hat der Kunde die Möglichkeit, eine Verbrauchskurve anzuzeigen (Verbrauch über die Zeit).

   Der Kunde hat außerdem die Möglichkeit, sich verschiedene Kennwerte berechnen zu lassen (z.B.: durchschnittlicher Verbrauch pro Tag im Monat x).

5. Der Kunde kontaktiert einen Energieberater oder schaltet ein Inserat am Portal für eine gewünschte Energieberatung
6. Ein Energieberater
   - wird auf das Inserat aufmerksam
   - wurde direkt vom Kunden angesprochen und reagiert auf die Anfrage
   - hat den Kunden über die Suchfunktion gefunden
7. Der Energieberater bittet den Kunden, ihm Daten zur Verfügung zu stellen.
8. Der Kunde gibt die gewünschten Daten an den Energieberater weiter.
9. Der Energieberater berechnet aus den übergebenen Daten ein Energiekonzept.
10. Der Energieberater sendet das Energiekonzept an den Kunden.

3.2.2 Ablauf aus Sicht des Energieberaters

1. Energieberater registriert sich und legt ein Nutzerprofil an
2. Energieberater sucht Kunden über:
   - ein Verzeichnis, geordnet nach Bundesland und Bezirk
   - ein Inserat eines Kunden, welcher eine Energieberatung wünscht
3. Der Energieberater setzt sich mit dem Kunden in Verbindung und fordert die notwendigen Daten an.
4. Der Kunde übergibt die notwendigen Daten an den Energieberater.
5. Der Energieberater berechnet aus den übergebenen Daten ein Energiekonzept.
6. Der Energieberater sendet das erstellte Energiekonzept an den Kunden.

3.2.3 Use Cases

Als nächstes wurde versucht, aus den groben Abläufen Use Cases für den Kunden, die automatischen Zähler des Kunden und den Energieberater herauszufiltern. Diese werden in Form eines Use-Case-Diagramms in Abbildung 3.2 zusammengefasst.

Beschreibung der Use Cases

Anlegen eines Kundendatensatzes
Auslösende Partei: Kunde
Voraussetzung: keine
Beschreibung: Der Kunde gibt deskriptive Daten zu seiner Person ein und speichert diese in Form eines Datensatzes im System. Zu diesen Daten gehören Name, Adresse,

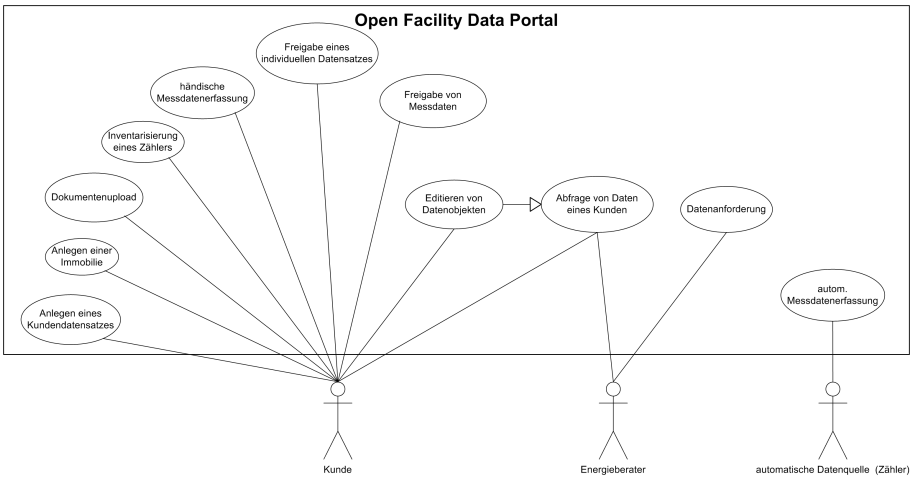


Abbildung 3.2: Use Case Diagramm für den Kunden, die automatischen Zähler des Kunden und den Energieberater

Email-Adresse,. . . .

Anlegen einer Immobilie
Auslösende Partei: Kunde
Voraussetzung: keine
Beschreibung: Der Kunde trägt eine neue Immobilie im System ein Dabei werden deskriptive Daten zur Immobilie gespeichert. Dazu gehören Adresse, Grundfläche, Nutzfläche, Art der Heizung, Anzahl der Geschoße,. . . .

Dokumentenupload
Auslösende Partei: Kunde
Voraussetzung: Falls das Dokument zu einer Immobilie gehört, muss diese zuerst angelegt werden.
Beschreibung: Der Kunde kann zu jeder seiner Immobilien oder unabhängig von Immobilien Dokumente hochladen, welche zur Energieberatung unterstützend freigegeben werden können. Hier wären beispielsweise Fotos der Immobilie, die letzten Energieabrechnungen oder der Energieausweis denkbar.

Inventarisierung eines Zählers
Auslösende Partei: Kunde
Voraussetzung: Die Immobilie, zu welcher der Zähler gehört, muss im System angelegt worden sein.
Beschreibung: Dieser Use Case dient zur Erfassung eines neuen Zählers des Kunden. Der Zähler wird einer bestimmten Immobilie des Kunden zugeordnet. Der Kunde kann bei der Inventarisierung wählen, ob es sich bei dem neuen Zähler um einen automatischen Zähler handelt oder ob er die Zählerstände händisch erfassen möchte.

Händische Messdatenerfassung
Auslösende Partei: Kunde
Voraussetzung: Der entsprechende Zähler wurde im System angelegt.
Beschreibung: Der Kunde liest manuell den Zählerstand einer seiner inventarisierten Zähler ab. Anschließend trägt er den abgelesenen Zählerstand im System händisch ein.

Freigabe eines individuellen Datensatzes
Auslösende Partei: Kunde
Voraussetzung: keine
Beschreibung: Der Kunde kann einem Energieberater einzelne Datensätze freigeben. Somit kann dieser ausgewählte Energieberater auf die freigegebenen Datensätze zugreifen und diese somit lesen. Bei individuellen Datensätzes handelt es sich um Kundeninformationsdatensätze, Dokumente oder Immobiliendatensätze.

Freigabe von Messdaten
Auslösende Partei: Kunde
Voraussetzung: keine
Beschreibung: Dieser Use Case beschreibt den Vorgang der Freigabe von Messdaten eines Kunden für einen ausgewählten Energieberater. Der Kunde wählt hierbei einen beliebigen Zeitraum aus, für welchen er die Messdaten einer Immobilie freigeben möchte. Der entsprechende Energieberater kann somit nach erfolgter Freigabe die Messdaten dieses Zeitraums einsehen und auswerten.

Abfrage von Daten eines Kunden
Auslösende Partei: Kunde oder Energieberater
Voraussetzung: keine
Beschreibung: Dieser Use Case kann sowohl vom Kunden selbst als auch von einem Energieberater ausgeführt werden. Er dient dazu, die Datensätze eines Kunden abzufragen und diese anschließend auszuwerten. Ein Kunde kann dabei alle seine Datensätze einsehen. Ein Energieberater kann allerdings nur jene Datensätze einsehen, welche der Kunde für ihn auch freigegeben hat.

Editieren von Datenobjekten
Auslösende Partei: Kunde
Voraussetzung: Der Datensatz wurde im System gespeichert und vom Kunden aufgerufen.
Beschreibung: Ein Kunde kann Datensätze, wie zum Beispiel Immobiliendatensätze oder den Kundeninformationsdatensatz, nachträglich ändern und anschließend aktualisiert speichert.

Datenanforderung
Auslösende Partei: Energieberater
Voraussetzung: Der Energieberater darf den Kunden kontaktieren.
Beschreibung: In diesem Use Case wählt der Berater jene Datensätze aus, welche er vom Kunden anfordern möchte. Dabei kann der Berater den Datensatz noch nicht im

Klartext sehen. Er sieht nur den Namen und die Art des Datensatzes.

Automatische Messdatenerfassung
Auslösende Partei: automatische Datenquelle (Zähler)
Voraussetzung: Der automatische Zähler wurde vom Kunden im System inventarisiert. Außerdem muss der Zähler richtig parametriert sein.
Beschreibung: In diesem Use Case liest die automatische Datenquelle den aktuellen Verbrauchsmesswert ab und speichert diesen im System.

3.3 Verschlüsselung und Weitergabe der Daten eines Kunden an den Energieberater

Da die Daten eines Kunden verschlüsselt gespeichert werden sollen, muss im Design berücksichtigt werden, wo die zur Verschlüsselung notwendigen Schlüssel generiert und aufbewahrt werden. Ein gangbarer Weg wäre eine Systemkomponente, welche diese Aufgabe für den Kunden übernimmt. Wenn dabei ein Verschlüsselungsalgorithmus eingesetzt wird, welcher asymmetrische oder hybride Verschlüsselung verwendet, so könnte die Weitergabe der Daten durch die Weitergabe der entsprechenden Schlüssel erfolgen. So darf der Schlüssel, welcher zur Verschlüsselung eingesetzt wird, nur vom Kunden abgefragt werden. Der Schlüssel zur Entschlüsselung hingegen darf einem bestimmten Berater freigegeben werden, damit auch dieser den Datensatz entschlüsseln kann.

Die Verschlüsselung der Daten dient dazu, dass der Betreiber der Datenbank, in welcher die Kundendaten gespeichert werden, die Daten selbst nicht lesen und somit in irgendeiner Form auswerten kann. Dies zeigt jetzt schon, dass die Komponente, welche die Schlüssel für den Kunden verwaltet, eine andere Komponente darstellt wie jene, in welcher die Daten gespeichert sind.

Ein weiterer Schritt wäre, alle Daten, welche zur Identifikation und zum Wiederauffinden von Datensätzen benötigt werden, ebenfalls zu verschlüsseln. Diese Art von Daten wird im Weiteren *Metadaten* genannt. Damit könnte der Anbieter des Datenspeichers nicht einmal die Art der Datensätze erkennen, welche ein Kunde gespeichert hat. Somit wäre es beispielsweise unmöglich zu erkennen, wie viele Immobilien oder Zähler ein Kunde angelegt hat.

Um dies zu ermöglichen, ist eine spezielle Art von Datensatz nötig, welcher diese Metadaten speichert. Allerdings ergibt sich durch die gemeinsame Speicherung der Metadaten mehrerer Datensätze in einem Metadatensatz ein Problem. Man nehme an, zwei Clients (des selben Kunden) bringen gleichzeitig Daten in das System ein. Dies könnte dann der Fall sein, wenn ein automatischer Zähler gerade einen neuen Messwert einträgt und der Kunde zeitgleich eine neue Immobilie anlegt. In diesem Fall müssten beide Clients ihren neuen Datensatz im selben Metadatensatz eintragen. Würde dies unkoordiniert von Statten gehen, könnte es hierbei zu Inkonsistenzen kommen. Das heißt, es könnte vorkommen, dass ein Client die Änderung des anderen Clients überschreibt und somit ein Eintrag verloren geht.

Um dies zu verhindern, muss ein Mechanismus geschaffen werden, welcher es ermöglicht, das Eintragen der Metadaten so zu koordinieren, dass Inkonsistenzen ausgeschlossen werden.

3.4 Komponenten des Systems

Wie eben schon erwähnt, besteht das System aus verschiedenen Komponenten. Abbildung 3.3 stellt diese Komponenten und deren Interaktion grafisch dar. Diese Komponenten sind die Schlüsselverwaltung, einer oder mehrere Datenspeicher, die automatischen Zähler des Kunden und ein Web-Portal, welches als User-Interface zwischen Kunden bzw. Beratern und dem System dient.

Nun soll noch kurz dargestellt werden, für welche Aufgaben diese Komponenten zuständig sind. In Kapitel 4.7 werden diese Komponenten und deren Aufgabenbereichte detailliert beschrieben.

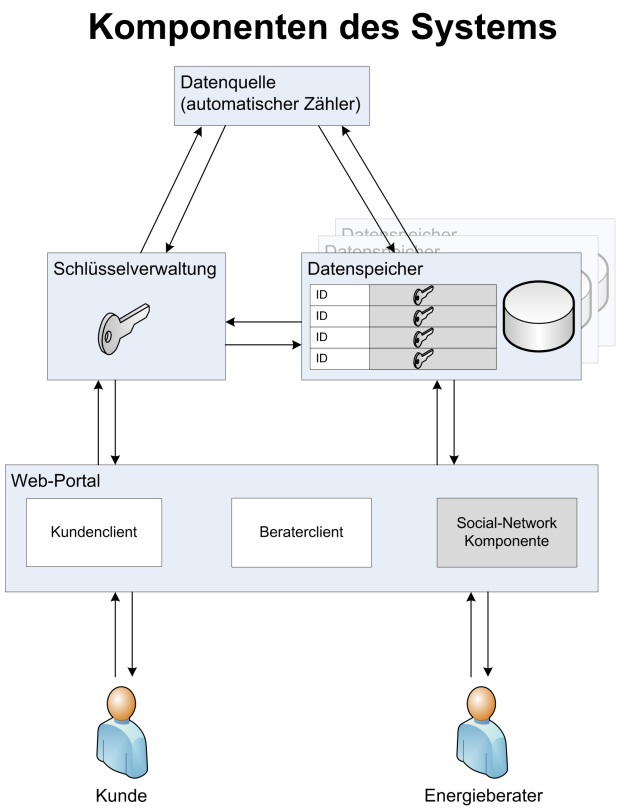


Abbildung 3.3: Komponenten des Systems

- ***Schlüsselverwaltung*:**

  - Erzeugung und Verwaltung der Schlüssel eines Kunden, welche zur Verschlüsselung der Daten eingesetzt werden
  - Erzeugung und Verwaltung der Signaturschlüssel der automatischen Zähler eines Kunden
  - Speicherung der Freigaben von ausgewählten Schlüsseln eines Kunden für bestimmte Energieberater

- ***Datenspeicher*:**
  - Speicherung der verschlüsselten Datensätzes eines Kunden

- ***automatischer Zähler* (auch *Datenquelle* genannt):**
  - automatische Zählerstandserfassung → Speicherung der verschlüsselten Zählerstände im Datenspeicher

- ***Web-Portal*:**
  - dient als User-Interface zwischen Kunde bzw. Energieberater und System → kann somit als Client-Applikation gegenüber den anderen Komponenten fungieren
  - beinhaltet eine Social-Network-Komponente, welche zur Kontakaufnahme zwischen Kunden und Energieberatern dient

3.5 Anforderungen an das Zielsystem

Am Ende der Analysephase wurden die Anforderungen an das Zielsystem genau definiert. Die folgende Auflistung soll diese Anforderungen zusammenfassen. Dabei soll das vorliegende Konzept vor allem die **fett** dargestellten Punkte erfüllen. Für alle anderen Punkte existieren Standardlösungen, auf welche an den entsprechenden Stellen im Designabschnitt (Kapitel 4) eingegangen wird.

- **Erfassung von Zählerständen der Kunden:**
  - **Stromzähler, Wasserzähler, Gaszähler, Fernwärmezähler, usw.**
  - **regelmäßige automatische Erfassung (holen des aktuellen Public-Keys und Verschlüsselung der Messdaten im Zähler)**
  - **händische Erfassung durch Eingabe des Kunden über ein Web-Interface**
  - **Speicherung der verschlüsselten Messwerte beim Hostingprovider (ein Kunde kann mehrere Zähler haben welche ihre Daten bei unterschiedlichen Hostingprovidern speichern)**
  - **ein Messdatensatz besteht aus Zeitstempel der Messwertermittlung, Messwert, Einheit des Messwerts, Public-Key-ID, User-ID**

- **Profil von Immobilien des Kunden**

  - **Beschreibung der Immobilien**
    - **Adresse, eventuell GPS-Koordinaten (geotagging)**
    - **Art (Wohnung, Wohnhaus, Büro, Firmengebäude,...)**
    - **Größe der Immobilie, Bauform, Isolierung,...**
    - **Energieverbrauch der letzten Jahre**
    - **Energiekosten der letzten Jahre**
  - **Möglichkeit zum Hochladen von weiteren Dateien (Energieausweis, Wärmebildmessungen,...)**
- **Kunde entscheidet, welche Daten immer sichtbar sind und welche erst auf Anfrage sichtbar werden**
- **Kunde kann jederzeit den aktuellen Zählerstand bzw. die Historie jedes Zählers einsehen (in tabellarischer Form und als Diagramm)**
- **alle Daten und Metadaten eines Kunden werden verschlüsselt gespeichert; der Datenspeicheranbieter (Hosting Provider) kann die Daten daher nicht auswerten**
- Kunde entscheidet ob er von einem Energieberater gefunden und kontaktiert werden darf
- Kunde kann sich, wenn er eine Beratung wünscht, auf eine „Inseratsliste“ setzen lassen um von Energieberatern leichter gefunden zu werden
- Energieberater können sich am Portal registrieren und ein Profil erstellen
  - Name, Adresse, Firmensitz, Bild (Foto, Firmenlogo,...), Link auf eigene Homepage
- Energieberater können Kunden suchen
  - über eine Liste von Kunden, welche gefunden und kontaktiert werden dürfen (Suchfunktion: Bundesland, Bezirk, Art des Objekts)
  - über eine Liste von „Inseraten“
- Kunden können Energieberater suchen
  - über eine Liste von Energieberatern (Suchfunktion: Bundesland, Bezirk)
- Dialogmöglichkeit zwischen Energieberater und Kunde (Mailbox)

3.6 Zusammenfassung

In diesem Abschnitt wurde die gestellte Aufgabe analysiert und erste Ideen zur Lösung gesammelt. Dazu wurde anfangs eine Mind Map erstellt. Danach wurde versucht die daraus gewonnene Information zu sortieren und in erste grobe Abläufe und Use Cases zu fassen. Anschließend wurden die Komponenten des Systems identifiziert und deren Aufgaben kurz beschrieben. Abschließend wurden daraus die konkreten Anforderungen an das Zielsystem definiert.

KAPITEL 4

Designkonzept

Im folgenden Kapitel wird das Designkonzept vorgestellt, welches die in Abschnitt 3.5 geforderten Anforderungen erfüllt. Dazu wird als erstes auf die automatische Zählerstandserfasssung eingegangen. Anschließend werden die verschiedenen Datensätze eines Kunden und deren Struktur vorgestellt. Danach wird beschrieben, wie und mit welchen Schlüsseln diese Daten vor der Speicherung verschlüsselt werden. Hiernach wird auf die verschlüsselte Speicherung der Metadaten eingegangen. Diese macht es erforderlich, dass ein Client für die Zeit der Speicherung exklusiven Zugriff auf die Daten des Kunden erhält, weshalb ein Lock-Mechnismus eingeführt wurde, welcher hier ebenfalls vorstellt wird. Danach werden die Aufgaben der in Abschnitt 3.4 identifizierten Komponenten definiert. Hiernach wird beschrieben, wie die Weitergabe der Daten eines Kunden an einen Energieberater abläuft. Danach werden die in Abschnitt 3.2.3 identifizierten Use Cases auf Systemprozesse abgebildet, deren Ablauf hier näher beschrieben wird. Anschließend werden die Interfaces zwischen den einzelnen Komponenten definiert. Am Ende des Kapitels wird eine Vorgangsweise zur historisierten Speicherung der Daten eines Kunden vorgestellt, welche es ermöglicht, den Datenbestand zu einem beliebigen Zeitpunkt in der Vergangenheit wiederherzustellen.

4.1 Automatische Messwerterfassung

In diesem Abschnitt soll beschrieben werden, wie die automatische Erfassung von Zählerständen funktioniert. Zu diesem Zweck kann jeder Kunde zu jeder seiner Immobilien, welche im System gespeichert sind, Zähler anlegen, welche automatisch regelmäßig Zählerstände in das System einbringen (z.B. alle 15 Minuten). Diese Zähler werden im weiteren auch als *automatische Datenquelle* bezeichnet. Das Anlegen eines neuen Zählers wird auch *Inventarisierung* genannt. Bei diesem Vorgang wird, wenn es sich um einen automatischen Zähler handelt, ein Parameterdatensatz angelegt, welcher alle notwendigen Daten enthält, die zur Einbringung der Messwerte notwendig

sind. Zu diesen Daten zählt, neben vielen anderen Daten, der Private-Key eines eigens für diesen Zähler genierten Signaturschlüsselpaares. Mit diesem Private-Key werden alle Requests, welche der Zähler an eine andere Komponente schickt, digital signiert. Diese Signatur dient der Integritäts- und der Authentizitätsprüfung des Requests durch die Gegenseite.

Messwerte, wie z.B. Zählerstände, werden von der autotomatischen Datenquelle zu Messwertdatensätzen serialisiert, welche anschließend mit wöchentlich wechselnden Schlüsseln verschlüsselt werden. Diese Schlüssel werden in der Schlüsselverwaltung generiert. Daher muss die Datenquelle regelmäßig aktuelle Schlüssel bei der Schlüsselverwaltung anfordern. Die Schlüssel werden wöchentlich gewechselt, da die Weitergabe der Messdaten so sehr einfach stattfinden kann. Hierbei erlaubt der Kunde dem Energieberater den Zugriff auf die Schlüssel (Private-Keys) jener Wochen, auf welche er dem Berater den Zugriff erteilen möchte.

Die folgende Ablaufbeschreibung beschreibt den prinzipiellen Ablauf bei der automatischen Messdatenerfassung, wenn das Gerät online ist:

1. Prüfung der Aktualität des Public-Keys (mithilfe einer Uhr in der Datenquelle) und eventuell Anforderung eines neuen Wochenschlüssels

2. Verschlüsselung des Messwertdatensatzes mithilfe des Public-Keys und Signatur mit dem Private-Key des Signaturschlüssels der Datenquelle. Diese Signatur dient der Integritätsprüfung und der Authentizitätsprüfung.

3. Versenden der Daten an jenen Datenspeicher, welcher im Gerät parametriert ist.

4. Beim Datenspeicher wird die Signatur auf Richtigkeit bzw. Unverfälschtheit geprüft.

5. Nach erfolgreicher Prüfung wird der Messwertdatensatz in der Datenbank des Datenspeichers gespeichert.

Da die Messdatenerfassung auch zuverlässig funktionieren muss wenn die Internetverbindung der Datenquelle nicht aufrecht gehalten werden kann, muss es in der Datenquelle einen Speicher geben, welcher Datensätze solange buffert, bis die Internetverbindung wieder hergestellt worden ist. Dabei werden die Datensätze solange verschlüsselt gespeichert, solange der Datenquelle noch aktuelle Wochenschlüssel zur Verfügung stehen. Danach werden alle Daten im Klartext in der Datenquelle gespeichert und erst verschlüsselt, wenn die Onlineverbindung wieder verfügbar ist und somit die notwendigen Schlüssel wieder angefordert werden können. Danach werden alle gebufferten Daten zur Speicherung an den Datenspeicher übertragen.

4.2 Daten eines Kunden

In diesem Abschnitt werden die verschiedenen Arten von Datensätzen, welche für einen Kunden im Datenspeicher abgelegt sind, vorgestellt. Zur besseren Übersicht können diese Daten eines Kunden als Baumstruktur dargestellt werden, welche als erstes vorgestellt wird (siehe dazu Abbildung 4.1):

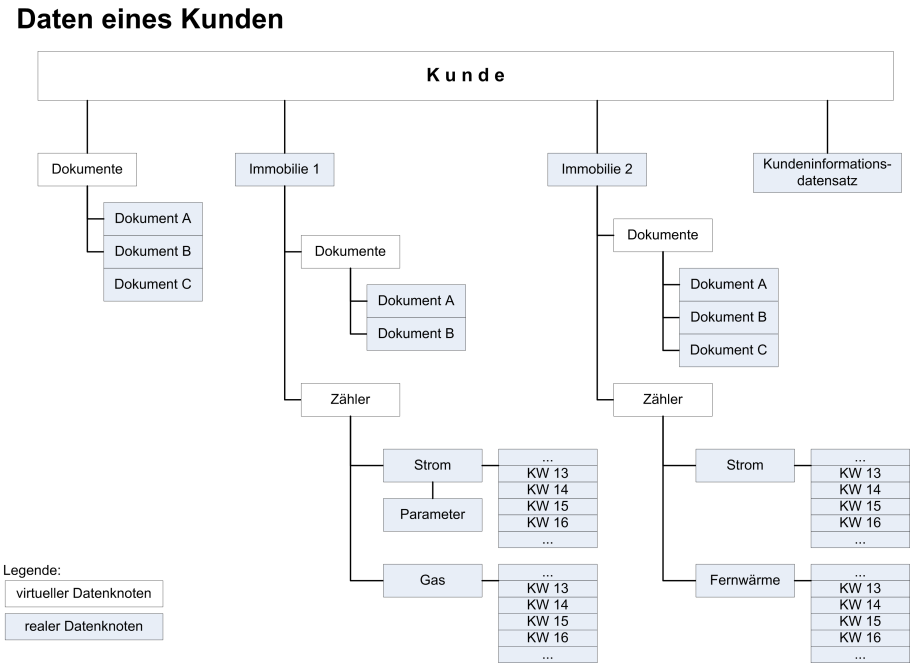


Abbildung 4.1: Baumstruktur der Daten eines Kunden

Es gibt drei verschiedene Arten von Datensätzen, welche dem Kunden direkt zugeordnet werden. Dazu gehört der Kundeninformationsdatensatz, welcher es dem Kunden ermöglicht, Daten zur Person, wie Name und Kontaktadresse, zu speichern. Weiters kann ein Kunde Dokumente hochladen, welche keiner Immobilie zuzuordnen sind, die aber trotzdem für die Energieberatung interessant sein können und somit einem Energieberater zur Verfügung gestellt werden können. Die dritte Art bilden die Immobiliendatensätze, in welchen nähere Daten zu den angelegten Immobilien des Kunden gespeichert werden. Zu diesen Daten zählen beispielsweise die Bauweise, die Art der Wärmedämmung, Größe der Immobilie, Art der Heizung, usw. Zu jeder Immobilie können nun Dokumente, wie beispielsweise der Energieausweis, Dokumente zu Wärmebildmessungen, die letzten Energieabrechnungen, usw. hochgeladen werden. Weitere Kindsknoten der Immobilien bilden die Zählerdatensätze, welche Informationen zu den invertarisierten Zählern, wie z.B. Zähler-ID, Name des Zählers, Art des Zählers, Messwerteinheit, usw. enthalten. Als Kindsknoten der Zählerdatensätze zählen einerseits der Parameterdatensatz (nur bei automatischen Zählern) und andererseits die Messwertdatensätze, welche die Zählerstände und die zugehörigen Zeitstempel der Ablesezeitpunkte enthalten. Bei händischen Zählern – das sind jene Zähler, welche vom Kunden selbst abgelesen und die Messwerte im Anschluss händisch ins System eingetragen werden – wird kein Parameterdatensatz erstellt und gespeichert.

Ein Kunde kann sein Datenaufkommen auf verschiedene Datenspeicher aufteilen. Dies ist allerdings nicht uneingeschränkt möglich. Bei allen Kundendokumenten, Im-

mobilien und beim Kundeninformationsdatensatz, kann der Kunde frei entscheiden, in welchem Datenspeicher er diesen Datensatz ablegen möchte. Alle Datensätze, welche zu einer bestimmten Immobilie zugeordnet sind, müssen hingegen im selben Datenspeicher wie der entsprechende Immobiliendatensatz gespeichert werden.

Nachdem nun die Baumstruktur der Daten eines Kunden vorgestellt wurde, werden nun die einzelnen Arten der Datensätze näher beschrieben. Abbildung 4.2 stellt diese grafisch dar. Dabei ist jeweils oben im weißen Feld der Name des Datensatzes eingetragen, welcher auch im entsprechenden Metadatensatz gespeichert wird. Die Metadaten dienen zur Identifikation bzw. zum Auffinden der Datensätze. Auf diese Thematik wird in Abschnitt 4.5 näher eingegangen.

In der Mitte der Datensätze sind in hellgrau die eigentlichen Datenfelder dargestellt. Alle Datensätze werden vor der Speicherung ins XML[1]-Format serialisiert und anschließend verschlüsselt. Dies soll durch das Schlüsselsymbol noch einmal verdeutlicht werden. Die entsprechenden XML-Schemas zur Serialisierung der Datensätze sind im Anhang A.3 zu finden.

Im dunkelgrauen Feld wird jeweils aufgelistet, mit welchen Schlüsseln ein Datensatz dieser Art üblicherweise verschlüsselt wird. Ein Datensatz kann dabei mit mehreren Schlüsseln (Public-Keys) gleichzeitig verschlüsselt sein, worauf im nächsten Abschnitt näher eingegangen wird. Es reicht allerdings ein Private-Key des entsprechenden Schlüsselpaares, um den Datensatz zu entschlüsseln und somit lesen zu können.

4.3 Verschlüsselung der Datensätze

Da die Anbieter der Datenspeicher die Datensätze der Kunden und die zugehörigen Metadaten aus Gründen der Privacy nicht lesen können sollen, müssen diese vor der Speicherung verschlüsselt werden. Die Verschlüsselung passiert schon beim Client des Kunden bzw. in der automatischen Datenquelle. Zur Verschlüsselung wird ein sogenanntes *hybrides* Verschlüsselungsverfahren (siehe Abschnitt 2.2.1) eingesetzt.

Bei der *hybriden* Verschlüsselung wird wie bei *asymmetrischen* Verschlüsselungsverfahren zur Ver- und Entschlüsselung ein Schlüsselpaar, bestehend aus einem Public-Key (Verschlüsselung) und einem Private-Key (Entschlüsselung) eingesetzt. Dabei werden die eigentlichen Daten symmetrisch mit einem sogenannten Session-Key verschlüsselt. Danach wird dieser Session-Key mithilfe eines asymmetrischen Verfahrens mit dem Public-Key des Schlüsselpaares verschlüsselt.

Bei der Entschlüsselung wird zuerst der verschlüsselte Session-Key mithilfe des Private-Keys asymmetrisch entschlüsselt. Anschließend werden die verschlüsselten Daten symmetrisch mit dem Session-Key entschlüsselt. Dieses Verfahren bietet nun neben den in Abschnitt 2.2.1 erwähnten Vorteilen eine weitere Möglichkeit, welche hier genutzt wird. Man kann mit diesem Verfahren einen Datensatz mit mehreren Public-Keys gleichzeitig verschlüsseln, wobei nur einer der entsprechenden Private-Keys ausreicht, um den Datensatz zu entschlüsseln. Um dies zu erreichen, wird einfach der Session-Key bei der Verschlüsselung des Datensatzes mit jedem einzelnen

[1] Extensible Markup Language

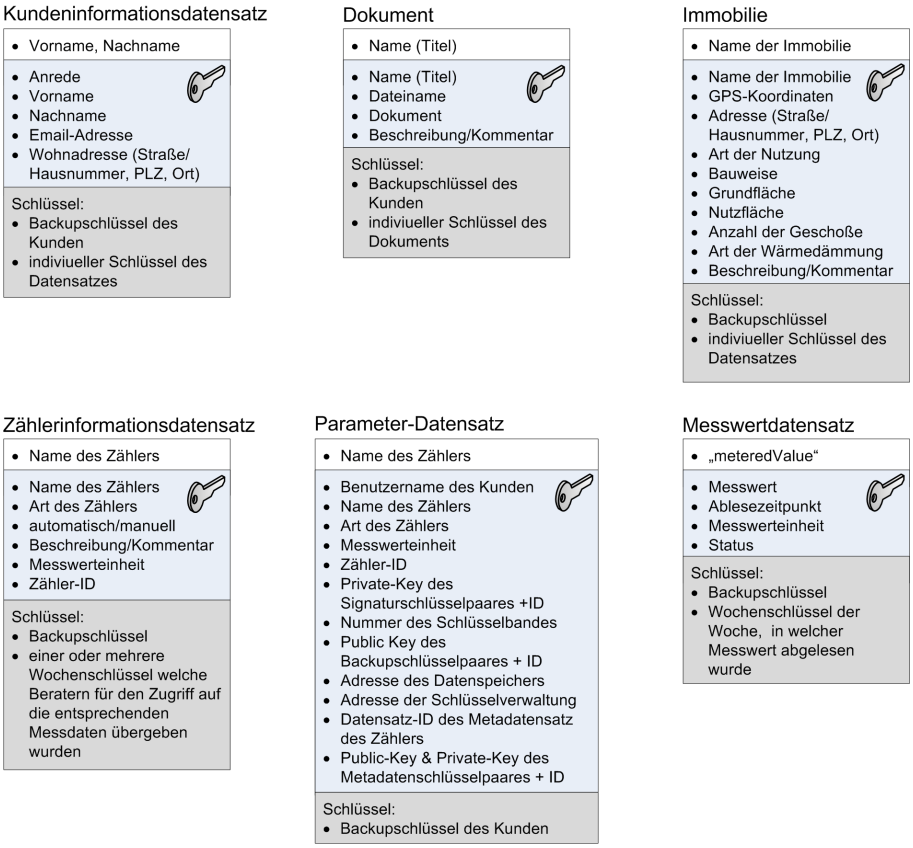


Abbildung 4.2: Aufbau der verschiedenen Datensätze eines Kunden

Public-Key verschlüsselt und jeder dieser verschlüsselten Session-Keys an den Datensatz angehängt. Nun kann jeder, der einen passenden Private-Key besitzt, den entsprechenden Session-Key entschlüsseln und somit in Folge auch den gesamten Datensatz. Der Vorteil ist nun, dass der Datensatz mit mehreren Schlüsseln gleichzeitig verschlüsselt werden kann, ohne aber die eigentlichen Daten mehrmals zu verschlüsseln oder gar mehrmals zu speichern. Es wird lediglich der Session-Key mehrmals verschlüsselt gespeichert. Dieser ist allerdings im Vergleich zu den Daten meist relativ klein. Dieser Trick wird auch bei der Verschlüsselung von Emails eingesetzt. Somit kann ein verschlüsseltes Email an mehrere Empfänger geschickt werden, ohne dass das eigentliche Email mehrmals verschlüsselt übertragen wird.

Im nächsten Abschnitt wird nun erklärt, welcher Nutzen aus der Verschlüsselung mit mehreren Schlüsseln gewonnen werden kann.

4.4 Schlüssel eines Kunden

Nachdem das verwendete Verschlüsselungsverfahren eingehend beschrieben wurde, werden in diesem Abschnitt die verschiedenen Schlüsselpaare eines Kunden und deren Verwendung genauer erklärt. In Abbildung 4.3 werden die verschiedenen Schlüsselpaare dargestellt. Die Schlüssel eines Kunden lassen sich in zwei Gruppen teilen: Signaturschlüsselpaare der automatischen Datenquellen und Schlüsselpaare zur Ver- und Entschlüsselung von Datensätzen. Zu der zweiten Gruppe gehören das Backup-Schlüsselpaar (eine Art Notfall oder Universalschlüssel des Kunden), die Wochenschlüsselpaare (zur Verschlüsselung der Messdaten), individuelle Datensatzschlüssel (zur Verschlüsselung von Immobilien-, Dokumenten-, und Kundeninformationsdatensätze) und das Metadaten-Schlüsselpaar (zur Verschlüsselung der Metadaten).

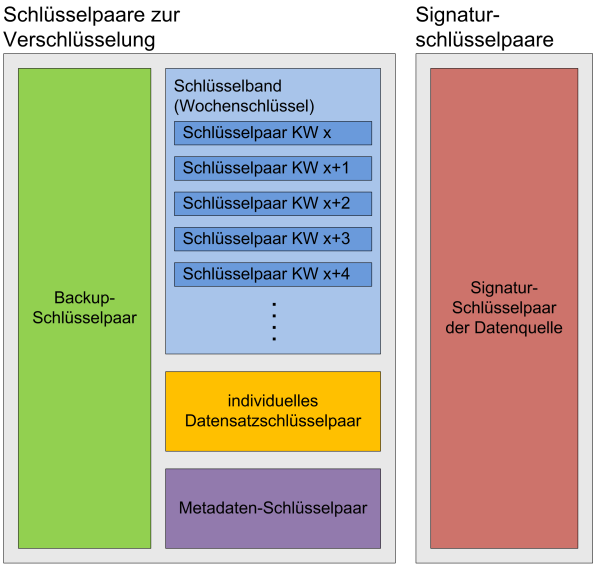


Abbildung 4.3: Übersicht über die Schlüssel eines Kunden

4.4.1 Backup-Schlüsselpaar

Anzahl der Schlüssel: ein Schlüssel pro Kunde
Zugriff auf den Public-Key: Auf den Public-Key des Backup-Schlüsselpaares hat ausschließich der Kunde selbst Zugriff.
Zugriff auf den Private-Key: Auf den Private-Key dieses Schlüsselpaares darf nur der entsprechende Kunde (Schlüsselbesitzer) zugreifen. Dieser Schlüssel kann nicht für andere Benutzer freigegeben werden.
Aufgabe des Schlüsselpaares: Das Backup-Schlüsselpaar eines Kunden dient als Notfallschlüssel und Universalschlüssel für alle Datensätze eines Kunden. Dieses Schlüsselpaar wird bei der Registrierung eines Kunden angelegt.

Jeder Datensatz, welcher für einen Kunden im Datenspeicher abgelegt wurde, ist zumindest mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Daher kann der Kunde mit dem Private-Key dieses Schlüsselpaares auch jeden seiner gespeicherten Datensätze entschlüsseln, was der Funktion eines Notfallschlüssels oder Universalschlüssels gleichkommt. Das heißt, der Datensatz kann auch dann noch entschlüsselt werden, wenn andere Schlüssel, wie z.B. ein Wochenschlüssel von Messdaten, verloren gehen. Dies hat zur Folge, dass der Backupschlüssel mit besonderer Sorgfalt aufbewahrt werden muss. Außerdem muss der Public-Key des Backup-Schlüsselpaares jeder Datenquelle, welche Daten für den Kunden im Datenspeicher ablegt, also speziell auch jedem automatischen Zähler, bekannt sein.
Art der damit verschlüsselten Datensätze: alle Datensätze eines Kunden

4.4.2 Schlüsselband (Wochenschlüssel)

Anzahl der Schlüssel: ein Schlüsselband pro angelegter Immobilie eines Kunden - ein Schlüsselpaar pro Woche für jedes Schlüsselband
Zugriff auf die Public-Keys: Auf die Public-Keys der Wochenschlüssel hat ausschließich der Kunde selbst Zugriff. Allerdings dürfen automatische Zähler eines Kunden die Public-Keys bei der Schlüsselverwaltung anfordern.
Zugriff auf die Private-Keys: Neben dem Kunden selbst haben auch vom Kunden ausgewählte Energieberater Zugriff auf die Private-Keys der Wochenschlüssel. Dazu muss der Kunde einzelnen oder allen Energieberatern den Zugriff auf bestimmte Wochenschlüssel gewähren (meist alle Private-Keys in einem gewissen Zeitraum).
Aufgabe des Schlüsselpaares: Beim Anlegen einer Immobilie durch einen Kunden wird ein sogenanntes Schlüsselband angelegt. Dieses beinhaltet Wochenschlüssel, welche zur Verschlüsselung von Messdaten dienen. Dies bedeutet, dass ein Messdatensatz immer mit dem Wochenschlüssel jener Kalenderwoche verschlüsselt wird, in welcher er ermittelt (abgelesen) wurde.

Durch Gewährung des Zugriffs auf die Private-Keys der Wochenschlüssel eines gewissen Zeitraumes, kann es Energieberatern ermöglicht werden, die Messdaten in diesem Zeitraum zu entschlüsseln. Das bedeutet, dass ein Berater, welcher Zugriff auf Private-Keys von Wochenschlüsseln hat, auch Zugriff auf alle Messdaten dieses Gebäudes in diesen Wochen hat.

Bei der Freigabe von Wochenschlüsseln wird zusätzlich der Zähler-Informationsdatensatz mit einem der freigegebenen Wochenschlüssel verschlüsselt. Dadurch erhält der Berater, für welchen die Schlüssel freigegeben werden, zusätzlich zu den Messdaten die Informationen zu den entsprechenden Zählern.
Art der damit verschlüsselten Datensätze: Messdatensätze, Zähler-Informationsdatensätze

4.4.3 Individuelles Datensatzschlüsselpaar

Anzahl der Schlüssel: ein Schlüssel für den Kundendatensatz, je ein Schlüssel für jede Immobilie eines Kunden und je ein Schlüssel für jedes hochgeladene Dokument eines Kunden (unabhängig davon, ob es sich um ein Kundendokument oder ein Dokument zu einer Immobilie handelt)

Zugriff auf den Public-Key: Auf den Public-Key dieses Schlüsselpaares hat ausschließich der Kunde selbst Zugriff.
Zugriff auf den Private-Key: Der Kunde kann einzelnen oder allen Energieberatern den Zugriff auf diesen Schlüssel erlauben, wenn er möchte, dass diese den damit verschlüsselten Datensatz lesen können. Daher hat sowohl der Kunde selbst, als auch ausgewählte oder alle Energieberater Zugriff auf diesen Schlüssel.
Aufgabe des Schlüsselpaares: Diese Art von Schlüsseln dient zur Verschlüsselung und Freigabe aller Datensätze, welche keine Messdaten beinhalten.
Art der damit verschlüsselten Datensätze: Kundeninformationsdatensatz, Immobilieninformationsdatensätze, Dokumente (Kundendokumente und Immobiliendokumente)

4.4.4 Metadaten-Schlüsselpaar

Anzahl der Schlüssel: ein Schlüssel pro Kunde
Zugriff auf den Public-Key: Auf den Public-Key des Metadaten-Schlüsselpaares hat ausschließich der Kunde selbst Zugriff.
Zugriff auf den Private-Key: Der Zugriff auf den Private-Key dieses Schlüsselpaares muss jedem Benutzer erteilt werden, welcher Zugriff auf irgendeinen Datensatzschlüssel (Private-Key eines Wochenschlüsselpaares oder eines individuellen Datensatzschlüsselpaares) hat.
Aufgabe des Schlüsselpaares: Dieses Schlüsselpaar dient zum Ver- und Entschlüsseln aller Metadaten-Datensätze des Kunden. Das heißt, dass jeder Berater, welcher Zugriff auf diesen Schlüssel hat, die Baumstruktur und die Namen der Datensätze eines Kunden, durch Entschlüsselung der Metadaten-Datensätze lesen kann. Sobald ein Kunde einem Berater einen Datensatzschlüssel freigibt, muss auch der Private-Key dieses Schlüsselpaares freigegeben werden.
Art der damit verschlüsselten Datensätze: alle Metadaten-Datensätze eines Kunden

4.4.5 Signaturschlüsselpaar der Datenquelle

Anzahl der Schlüssel: ein Schlüssel pro automatischer Datenquelle eines Kunden
Zugriff auf den Public-Key: Neben dem Kunden kann jeder Datenspeicher auf den Public-Key dieses Schlüsselpaares zugreifen
Zugriff auf den Private-Key: Auf den Private-Key dieses Schlüsselpaares darf nur der Kunde zugreifen.
Aufgabe des Schlüsselpaares: Bei der Inventarisierung eines automatischen Zählers wird für den Kunden ein neues Signatur-Schlüsselpaar generiert. Der Private-Key kann im Anschluss gemeinsam mit den anderen Parameterdaten heruntergeladen und im Zähler parametriert werden. Jeder Request, welcher von einem automatischen Zähler ausgesendet wird, wird mit Hilfe des Private-Keys digital signiert. Der Datenspeicher und die Schlüsselverwaltung können anschließend mithilfe des entsprechenden Public-Keys die Signatur überprüfen. Dadurch wird garantiert, dass die Daten bei der Übertragung nicht verfälscht wurden (Integritätsprüfung) und von einer authorisierten Quelle stammen.

Art der damit signierten Daten: Alle Requests, welche eine automatische Datenquelle an eine andere Komponente verschickt.

4.5 Speicherung der Metadaten

Um zu verhindern, dass Datenspeicheranbieter (Hosting-Provider) durch Auswertung der Daten, welche zum Wiederauffinden und zur Identifikation der eigentlichen Datensätze notwendig sind, unbefugt Informationen gewinnen können, werden diese *Metadaten* ebenfalls verschlüsselt gespeichert. Würden diese Metadaten nicht verschlüsselt gespeichert werden, könnte der Datenspeicheranbieter zumindest herausfinden, wieviele Immobilien und zugehörige Zähler ein Kunde gespeichert hat. Dies allein könnte allerdings schon zu gezielten Werbezwecken eingesetzt werden.

Um ein solches Vorgehen zu verhindern, werden die Metadaten, wie z.B. die ID, der Name, die Art des Datensatzes, in einer speziellen rekursiven Datenstruktur namens *Metadata* gespeichert, welche anschließend serialisiert und verschlüsselt werden kann. Diese Datenstruktur ermöglicht es die Baumstruktur der Kundendaten wiederzuspiegeln. Abbildung 4.4 zeigt das Klassendiagramm dieser Datenstruktur.

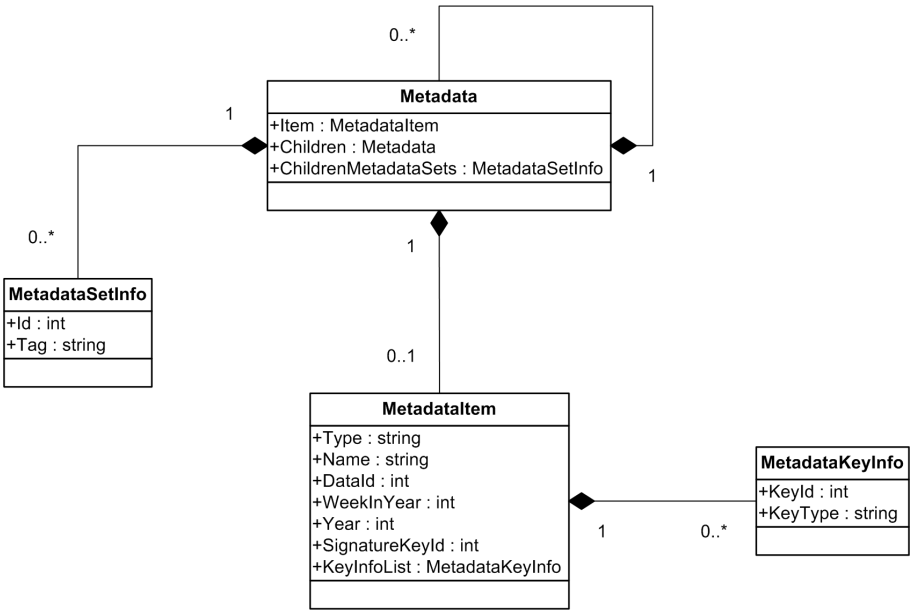


Abbildung 4.4: Klassendiagramm der Typen *Metadata*, *MetadataItem*, *MetadataSetInfo* und *MetadataKeyInfo*

Diese Datenstruktur ist folgendermaßen aufgebaut: Jedes Objekt vom Typ *Metadata* enthält ein *Item* vom Typ *MetadataItem*. In diesem werden die Metadaten des zugehörigen Datensatzes gespeichert. Dazu gehören der Typ, der Name und die ID des Datensatzes. Sollte es sich bei dem Datensatz um einen Messwertdatensatz handeln, so wird zusätzlich die Kalenderwoche und das Jahr gespeichert. Wurde dieser

Datensatz durch eine automatische Datenquelle eingebracht, so wird hier auch die Schlüsselpaar-ID des Signaturschlüssels der Datenquelle gespeichert. Darüber hinaus ist die sogenannte *KeyInfoList* vom Typ *MetadataKeyInfo* hier gespeichert. Diese speichert die Informationen zu den Schlüsseln, mit welchen der Datensatz verschlüsselt wurde. Dazu gehört die Schlüssel-ID des entsprechenden Schlüsselpaares und der Typ des Schlüssels (also z.B. *backup*, *individual*,...).

Neben dem soeben beschriebenen Item enthält der Typ *Metadata* eine Liste von *Children*, welche wiederum vom Typ *Metadata* sind (aufgrund dieser ist die Datenstruktur rekursiv). Für Kinder-Metadatensätze, welche in einem gesonderten Metadata-Datensatz gespeichert sind, kann über die Liste *ChildrenMetadataSets* ein entsprechender Verweis auf diesen Datensatz eingefügt werden. Dieser ist vom Typ *MetadataSetInfo* und enthält neben der ID des Metadata-Datensatzes auch ein *Tag*, über welches der Verweis zum einfacheren Wiederauffinden markiert werden kann.

Zur Speicherung der Metadaten im Datenspeicher, kann die gezeigte Datenstruktur ins XML-Format serialisiert und anschließend verschlüsselt werden. Das entsprechende XML-Schema findet man im Anhang A.4.

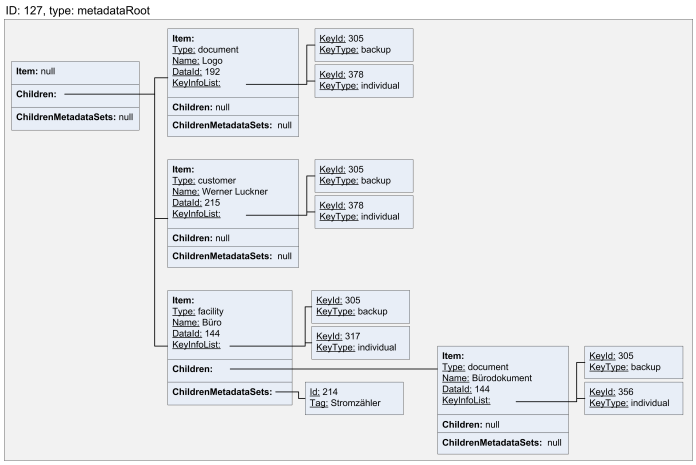


Abbildung 4.5: Anwendungsbeispiel der Metadaten-Datenstruktur (Teil 1)

Nun soll anhand eines einfachen Beispiels gezeigt werden, wie die soeben gezeigte Datenstruktur eingesetzt wird, um die Metadaten der Datensätze eines Kunden zu speichern. Dabei wurde versucht, zum einen die Metadatensätze nicht allzu groß werden zu lassen und zum anderen möglichst wenige Metadatensätze zu verwenden, da viele kleine Datensätze bei der Ver- und Entschlüsselung weniger effizient sind als wenige große. Ein gemeinsamer Metadatensatz für alle Metadaten wäre nicht ideal, da dieser möglicherweise sehr groß werden würde und somit die Serialisierung sehr speicherintensiv und langsam sein könnte. Die Abbildungen 4.5 und 4.6 stellen das Beispiel grafisch dar. Alle nicht verwendeten Attribute wurden der Übersichtlichkeit halber weggelassen.

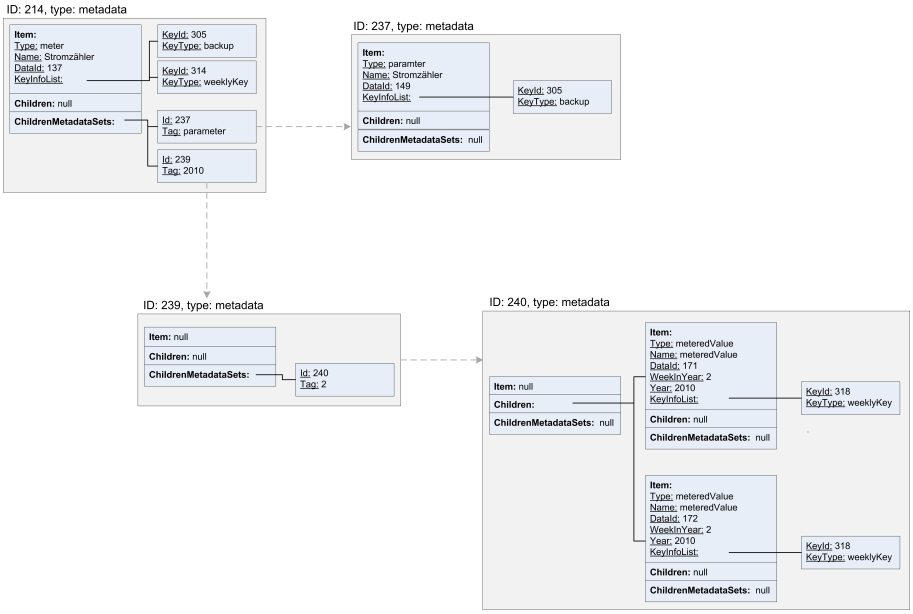


Abbildung 4.6: Anwendungsbeispiel der Metadaten-Datenstruktur (Teil 2)

Der in Abbildung 4.5 abgebildete *Metadata*-Datensatz enthält die Wurzel der Metadaten des Kunden im entsprechenden Datenspeicher. Dieser Datensatz wird zum einfacheren Auffinden mit dem *Typ metadataRoot* im Datenspeicher abgelegt. Alle anderen Metadatensätze werden mit dem *Typ metadata* gespeichert. Die Wurzel selbst verweist auf keinen Datensatz. Daher hat deren *Item* den Wert *null*. Darüber hinaus hat die Wurzel drei verschiedenartige *Children*-Datensätze. Beim ersten handelt es sich um einen Metadatensatz für das Kundendokument „Logo“, welches sowohl mit dem Backup-Key, als auch mit einem individuellen Datensatzschlüssel verschlüsselt ist. Beim nächsten Datensatz handelt es sich um die Metadaten für den Kundeninformationsdatansatz. Als drittes ist ein Metadatensatz für einen Immobiliendatensatz zu sehen. Dieser hat zwei Kinder, wobei sich eines, das Immobiliendokument „Bürodokument“, direkt in diesem Metadatensatz befindet. Das andere Kind, mit dem *Tag* „Stromzähler“, ist in einen eigenen Metadatensatz ausgelagert.

Dieser hat die Datensatz-ID „214“ und ist auf Abbildung 4.6 links oben zu sehen. Dieser Metadatensatz besitzt wiederum zwei Kinder, welche wiederum in eigene Metadatensätze ausgelagert wurden. Jener mit der ID „237“ beinhaltet die Metadaten des Parameterdatensatzes des Zählers. Der andere Metadatensatz, auf welchen mit dem *Tag* „2010“ verwiesen wird (ID „239“), verweist selbst auf keinen Datensatz, sondern beinhaltet nur die Liste von Verweisen aller Metadatensätze der verschiedenen Wochen des Jahres, wobei der *Tag* für die entsprechende Kalenderwoche steht. Der Metadatensatz mit der ID „240“ beinhaltet nun alle Metadaten jener Messdatensätze, welche zur Kalenderwoche „2“ im Jahr „2010“ gehören.

Die in Kapitel 5 vorgestellte Demo-Applikation verwendet genau die Aufteilung der Metadaten des hier gezeigten Beispiels.

Die Metadatensätze sind allesamt mit dem Public-Key des Backup-Schlüsselpaares und mit dem Public-Key des Metadata-Schlüsselpaares verschlüsselt. Wann immer ein Kunde einem Berater Daten freigeben möchte, muss neben dem Private-Key des Datensatzschlüsselpaares auch der Private-Key des Metadatensatzschlüsselpaares freigegeben werden.

4.6 Der Lock-Mechanismus

Da die Metadaten der Datensätze verschlüsselt gespeichert werden, ist es notwendig, dass Clients, welche neue Daten einbringen oder Datensätze aktualisieren, sich für die Zeit zwischen dem Zeitpunkt, an dem der entsprechende Metadatensatz zur Aktualisierung gelesen wird und dem Zeitpunkt an dem alle neuen oder aktualisierten Datensätze im Datenspeicher gesichert sind, exklusiven Schreibzugriff auf die Datensätze des Kunden haben. Zu diesem Zweck wurde ein Mechanismus eingeführt, welcher sicherstellt, dass zu jedem Zeitpunkt pro Kunde nur ein Client schreibend auf einen Datenspeicher zugreift. Sollte der Fall eintreten, dass zwei oder mehrere Clients eines Kunden auf den selben Datenspeicher schreibend zugreifen möchten (das ist z.B. dann der Fall, wenn ein Kunde ein Dokument hochlädt und gleichzeitig einer seiner automatischen Zähler einen Messwert eintragen möchte), so wird sichergestellt, dass zu jedem Zeitpunkt nur ein Client das Lock besitzt und somit den alleinigen (schreibenden) Zugriff auf den Datenspeicher erlangt. Alle anderen Clients, werden solange verzögert, bis der Client, welcher das Lock gerade besitzt, dieses Lock wieder löst. Danach erhält der nächste wartende Client das Lock (nach dem FIFO-Prinzip).

Sollte ein Client, welcher gerade im Besitz des Locks ist, aus irgendeinem Grund das Lock nicht mehr lösen (weil dieser beispielsweise abgestürzt ist), so verhindert ein Timeout, dass dieser Client das Lock für immer blockiert. Sollte die Timeout-Zeit verstrichen sein, ohne dass der Client des Kunden schreibend auf den Datenspeicher zugegriffen hat oder das Lock gelöst wurde, so wird das Lock automatisch gelöst und an den nächsten wartenden Client weitergegeben. Sollte der Client, welcher sich zu lange nicht gemeldet hat, nun nach Ablauf der Timeout-Zeit einen schreibenden Zugriff auf den Datenspeicher durchführen wollen, so wird dieser abgelehnt und der Client über diesen Fehler informiert.

Ein Client, welcher das Lock erhält, empfängt einen Lockstring, welcher das aktuell aktive Lock eindeutig identifiziert.

4.6.1 Anforderung des Locks

Zur Anforderung des Locks sendet der Client einen *SubscribeLockRequest* an den Datenspeicher. Nun können, je nachdem, ob das Lock gerade frei ist oder nicht, verschiedene Szenarien eintreten (Abbildung 4.7 stellt die beiden Situationen grafisch

dar):

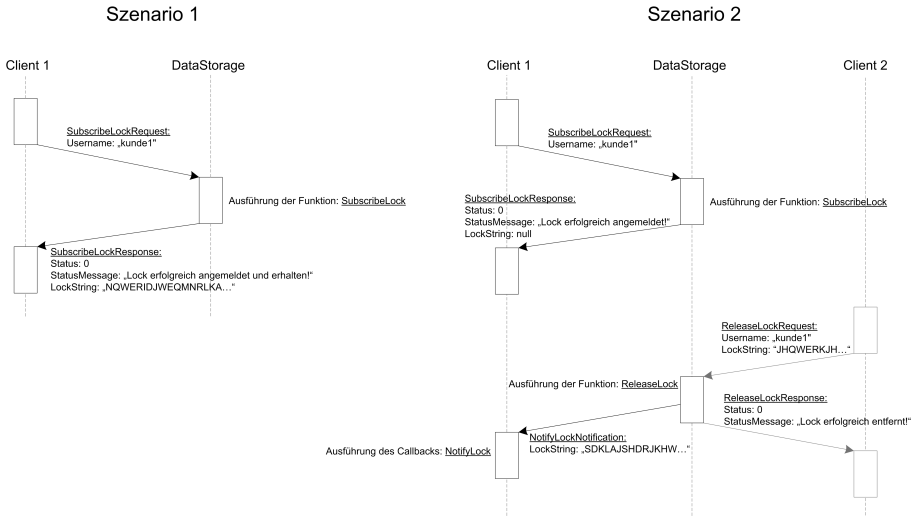


Abbildung 4.7: Anforderung des Locks

Szenario 1 - das Lock des Kunden ist *frei*
Wenn das Lock des Kunden frei ist wenn der Request verarbeitet wird, so erhält der Client sofort das Lock. Daher beinhaltet der *SubscribeLockResponse* den neu erzeugten Lockstring.

Szenario 2 - das Lock des Kunden ist *nicht frei*
Sollte das Lock des Kunden zum Zeitpunkt der Verarbeitung des Requests nicht frei sein (ein andere Client hält das Lock), so enthält der *SubscribeLockResponse* keinen Lockstring (das entsprechende Attribut hat den Wert *null*). Nachdem der Client, welcher das Lock gerade hält (*Client 2* in Abbildung 4.7), das Lock wieder freigibt (oder die Timeout-Zeit abläuft), so wird an den Client, welcher auf das Lock wartet (*Client 1* in Abbildung 4.7), ein Callback geschickt, welcher den Lockstring enthält.
Hinweis: Da Callbacks nicht von allen Plattformen unterstützt werden, kann der Mechanismus natürlich sehr einfach auf einen Mechanismus umgestellt werden, welcher anstatt der Callbacks Polling einsetzt. Das bedeutet, dass wartende Clients regelmäßig beim Datenspeicher anfragen ob das Lock, auf das sie gerade warten, schon verfügbar ist.

Ein weiteres Problem, welches bei Callbacks berücksichtigt werden muss, ist, dass aufgrund des Rückkanals der Client einen Port braucht, welcher vom Server aus erreichbar sein muss. Dies bedeutet, dass dieser Port bei einer eventuellen Firewall durchgeschaltet sein muss.

Die Abarbeitung der wartenden Locks eines Kunden erfolgt nach dem FIFO-Prinzip. Dies bedeutet, dass die Clients in der Reihenfolge das Lock bekommen, in welcher sie das Lock angefordert haben.

4.7 Komponenten des Systems und deren Interaktion

Nachdem nun alle Grundlagen des Systems beschrieben wurden, soll der folgende Abschnitt, die in Abschnitt 3.4 vorgestellten Komponenten detailliert beschreiben. Darüber hinaus kann der Aufgabenbereich der einzelnen Komponenten genau abgesteckt werden. Zur besseren Vorstellung der Interaktion zwischen den Komponenten sei auf Abbildung 3.3 verwiesen.

4.7.1 Automatischer Zähler (Datenquelle)

Automatische Zähler haben die Aufgabe, den Messwert (Zählerwert) zu erfassen und diesen anschließend an den Datenspeicher zu übertragen. Bevor die Daten an den Datenspeicher übertragen werden, müssen diese mit dem aktuellen Wochenschlüssel des Schlüsselbandes der Immobilie, für welche der Zähler inventarisiert ist, verschlüsselt werden. Sollte der aktuelle Wochenschlüssel noch nicht in der Datenquelle vorhanden sein, muss dieser bei der Schlüsselverwaltung angefordert werden. Zusätzlich zur Verschlüsselung wird jeder Datensatz, welcher zur Speicherung an den Datenspeicher geschickt wird, digital signiert. Zu diesem Zweck ist ein Private-Key eines Signaturschlüsselpaares in der Datenquelle parametriert. Die Signatur dient einerseits zur Überprüfung der unverfälschten Übertragung der Messdaten und andererseits zur Authentifizierung der Datenquelle.

Da es vorkommen kann, dass der Datenquelle nicht jederzeit eine Verbindung zum Internet zur Verfügung steht, müssen Messdaten, welche noch nicht an den Datenspeicher übertragen wurden, in der Datenquelle zwischengespeichert werden. Dazu werden Messdatensätze, für welche noch aktuelle Wochenschlüssel vorhanden sind, verschlüsselt gespeichert. Diese sind dann quasi schon versandfertig. Messdatensätze, für welche keine aktuellen Wochenschlüssel mehr vorhanden sind (weil die Online-Verbindung zur Schlüsselverwaltung fehlt), werden bis zur Wiederherstellung der Internetverbindung unverschlüsselt in der Datenquelle zwischengespeichert. Sobald die Datenquelle die Verbindung zum Internet wiederhergestellt hat, werden die fehlenden Wochenschlüssel bei der Schlüsselverwaltung angefordert und die noch nicht verschlüsselten Messdatensätze verschlüsselt. Anschließend werden diese Datensätze ebenfalls signiert und zusammen mit den anderen schon verschlüsselten und signierten Datensätzen zum Datenspeicher zur Speicherung übertragen. Außerdem muss die automatische Datenquelle vor dem Senden der Messdatensätze das in Abschnitt 4.6 beschriebene Lock anfordern und anschließend die Messwertdatensätze in die entsprechenden Metadatensätze eintragen.

4.7.2 Schlüsselverwaltung

Die Schlüsselverwaltung generiert, speichert und verwaltet die Schlüsselpaare, welche zur Signatur und zur Ver- bzw. Entschlüsselung der Daten verwendet werden. Für jeden Benutzer des Systems wird bei der Registrierung ein sogenanntes Backup-Schlüsselpaar generiert, welches zum einen als Notfallschlüssel und zum anderen als Universalschlüssel dient. Da mit diesem Schlüssel jeder Datensatz eines Benutzers verschlüsselt ist, kann auch jeder Datensatz mit diesem Schlüssel entschlüsselt und als Klartext dargestellt werden. Neben dem Backup-Schlüsselpaar werden je nach Art der Daten eines Kunden verschiedene Schlüssel generiert und für den Kunden gespeichert. Dazu gehören beispielsweise individuelle Datensatzschlüsselpaare, welche zur Verschlüsselung von Dokumenten, des Kundeninformationsdatensatzes und von Immobilieninformationsdatensätzen verwendet werden und Schlüsselbänder mit Wochenschlüsselpaaren, welche zur Ver- bzw. Entschlüsselung von Messdatensätzen verwendet werden. Ein weiteres Schlüsselpaar bildet das Metadaten-Schlüsselpaar, welches für die Ver- und Entschlüsselung der Metadaten des Kunden zuständig ist.

Außer dem Private-Key des Backup-Schlüsselpaares können alle Private-Keys jener Schlüsselpaare, welche für die Ver- und Entschlüsselung verwendet werden, für den Zugriff durch andere Benutzer freigegeben werden. Wenn ein Schlüssel für einen anderen Benutzer freigegeben wurde, kann dieser Benutzer diesen Schlüssel lesen und somit alle mit diesem Schlüssel verschlüsselten Daten entschlüsseln. Diese Berechtigungen, also wer auf welche Private-Keys eines Kunden zugreifen darf, werden ebenfalls in der Schlüsselverwaltung gespeichert. Der Kunde kann dabei wählen, ob er den Zugriff auf einen Schlüssel einem bestimmten oder allen anderen Benutzern erlaubt. Wann immer ein Kunde einem anderen Benutzer den Zugriff auf einen seiner Datensatzschlüssel erlaubt, muss auch der Private-Key des Metadaten-Schlüsselpaares freigegeben werden.

Neben den Schlüsseln zur Ver- und Entschlüsselung der Daten, werden in der Schlüsselverwaltung auch noch Schlüsselpaare zur Signatur der Requests von automatischen Datenquellen generiert und verwaltet. Dazu wird bei der Inventarisierung eines Zählers im System ein Signaturschlüsselpaar erzeugt, wessen Private-Key im automatischen Zähler parametriert wird. Dieser automatische Zähler signiert jeden Request, welchen er an einen Datenspeicher oder an die Schlüsselverwaltung sendet, mit diesem Schlüssel. Der Datenspeicher bzw. die Schlüsselverwaltung kann beim Empfang mithilfe des entsprechenden Public-Keys die Gültigkeit der Signatur überprüfen.

Die Schlüsselverwaltung spielt eine zentrale Rolle im System und kommt daher im System nur einmal vor.

4.7.3 Datenspeicher

Der Datenspeicher ist für die Aufbewahrung und Zurverfügungstellung der Daten eines Kunden zuständig. Um ein unerwünschtes Lesen von Daten durch Unbefugte oder ein Verwerten der Daten durch den Betreiber des Datenspeichers (Hosting-Provider) zu verhindern, werden alle Daten und Metadaten, welche im Datenspeicher abgelegt sind, ausschließlich in verschlüsselter Form gespeichert. Dies bedeutet, dass von jedem Datensatz nur jene Daten unverschlüsselt lesbar sind, welche zum Wie-

derauffinden und Zuordnen des Datensatzes notwendig sind. Unverschlüsselt wird daher nur die ID des Datensatzes gespeichert und ob es sich bei dem Datensatz um einen Metadatensatz oder einen normalen Datensatz handelt. Die eigentlichen Daten werden verschlüsselt gespeichert. Zur Verschlüsselung wird ein hybrides Verschlüsselungsverfahren eingesetzt (siehe Abschnitt 4.3). Durch dieses Verfahren ist es möglich, einen Datensatz gleichzeitig mit mehreren Public-Keys zu verschlüsseln. Zur Entschlüsselung reicht lediglich einer dieser Schlüssel. Jeder Datensatz ist dabei, unabhängig von der Art des Datensatzes, zumindest mit dem Public-Key des Backup-Schlüsselpaares des Kunden verschlüsselt.

Das System schränkt die Anzahl der Anbieter eines Datenspeichers nicht ein. Dies bedeutet, dass es im System mehrere Datenspeicher geben kann. Darüber hinaus kann ein Kunde sein Datenaufkommen auf mehrere Datenspeicher aufteilen. Dabei ist allerdings zu beachten, dass alle zu einer Immobilie gehörenden Daten in ein und demselben Datenspeicher abgelegt werden müssen. Das heißt, ein Kunde kann beispielsweise alle Daten von Immobilie A und Immobilie B bei Datenspeicheranbieter X speichern und alle Daten von Immobilie C bei Datenspeicheranbieter Y.

Der Datenspeicher bietet die Möglichkeit zum Einbringen, Bearbeiten und Abfragen von Daten und Metadaten. Dabei darf nur der Kunde selbst Daten einbringen und bearbeiten. Das Abfragen der Daten ist hingegen auch durch andere Benutzer möglich. Es entsteht dadurch kein Sicherheitsrisiko, da alle Daten verschlüsselt gespeichert werden. Dies bedeutet, dass nur jene Benutzer, welche auch Zugriff auf die entsprechenden Private-Keys haben die Daten wirklich lesen können. Zusätzlich muss bei der Abfrage die Schlüssel-ID eines Schlüssels, mit welchem der gewünschte Datensatz entschlüsselt werden kann, angegeben werden. Zusätzlich bietet der Datenspeicher die in Abschnitt 4.6 beschrieben Funktionen des Lock-Mechanismus, welcher dafür zuständig ist, dass die Metadaten konsistent gehalten werden können.

4.7.4 Web-Portal (inkl. Clientkomponenten)

Das Web-Portal dient als User-Interface und Client für die Benutzer. Außerdem stellt diese Komponente eine Möglichkeit zur Kontaktaufnahme zwischen Energieberatern und Kunden zur Verfügung (siehe Abschnitt 4.8). Das Web-Portal kann auch als Bindeglied zwischen Schlüsselverwaltung, Datenspeicher und den Benutzern gesehen werden. Es übernimmt sowohl für den Kunden, als auch für den Energieberater die Rolle des Clients hinsichtlich der Komponenten *Datenspeicher* und *Schlüsselverwaltung*. Es fordert für den Benutzer die gewünschten Schlüssel zur Ver- und Entschlüsselung der Daten bzw. Metadaten bei der Schlüsselverwaltung an, ver- bzw. entschlüsselt diese, bereitet diese für die Darstellung auf und stellt diese in Form einer Web-Seite dar. Außerdem kann der Kunde über das Web-Portal neue Datensätze in das System einbringen. Er kann beispielsweise neue Immobilien anlegen, Dokumente hochladen oder Messwerte von manuellen Zählern erfassen. Außerdem kann der Kunde über das Web-Portal die im Datenspeicher abgelegten Datensätze editieren und um weitere Informationen ergänzen. Die Clientkomponenten spielen außerdem eine sehr wichtige Rolle bei der Konsistenzhaltung der Daten bzw. Metadaten. Auf dieses Thema wird in Abschnitt 4.6 und in Abschnitt 6.1 ausführlich eingegangen.

Ein wichtiger Bestandteil des Web-Portals bildet eine Social Network Komponente, welche in Abschnitt 4.8 näher beschrieben wird.

4.8 Kontaktaufnahme zwischen Kunden und Energieberatern

Wie im vorherigen Abschnitt schon erwähnt, beinhaltet das Web-Portal eine Social-Network-Komponente, welche zur Kontaktaufnahme zwischen Kunden und Beratern dient. Diese Komponente erlaubt es einem Kunden, der eine Beratung wünscht, an einen Energieberater heranzutreten. Umgekehrt soll es auch für Energieberater möglich sein, Kunden, welche beraten werden wollen, zu suchen und anschließend zu kontaktieren. Um von einem Energieberater kontaktiert werden zu dürfen, muss der Kunde vorher seine Zustimmung erteilen. Außerdem könnte diese Komponente eine Art Mailbox beinhalten, welche den direkten Austausch von Nachrichten zwischen Kunden und Beratern ermöglicht.

Kunden, welche von Beratern, mit welchen sie über das Portal noch nicht in Kontakt stehen, nicht kontaktiert werden dürfen, können einem Berater ein sogenanntes Token zukommen lassen, welches es dem Berater erlaubt, den Kontakt mit dem Kunden über das Portal aufzunehmen. Dieses Token stellt sozusagen eine Ausnahmegenehmigung für das Kontaktieren durch unbekannte Personen dar. Dies könnte beispielsweise dann hilfreich sein, wenn der Kunde mit dem Energieberater schon außerhalb des Portals Kontakt aufgenommen hat und diesen Kontakt nun auch über das Portal pflegen möchte. Die Form dieses Tokens kann je nach Implementierung sehr verschieden sein. Denkbar wäre es in Form einer URL oder eines Codes, welchen sich der Kunde ausdrucken kann, um ihn dem Berater zu übergeben. Dieser Code oder diese URL identifiziert anschließend eindeutig die Erlaubnis, diesen Kunden kontaktieren zu dürfen.

Darüber hinaus könnte diese Komponente außerdem eine Art Forum beinhalten, in welchem sich die Benutzer untereinander über ihre Erfahrungen austauschen und Probleme besprechen. Beispielsweise könnten Kunden berichten, durch welche Maßnahmen sie wieviel Energie eingespart haben. Das heißt, dass durch das Web-Portal nicht nur eine Kommunikation zwischen Kunden und Energieberatern ermöglicht wird, sondern auch die Kommunikation von Kunde zu Kunde.

Diese Social-Network Komponente erfüllt somit alle in Abschnitt 3.5 nicht fett markierten Punkte.

4.9 Weitergabe der Daten eines Kunden

Nachdem die Aufgabenbereiche der einzelnen Komponenten genau beschrieben und abgegrenzt wurden, kann in diesem Abschnitt darauf eingegangen werden, wie die Weitergabe von Datensätzen funktioniert.

Wenn ein Kunde einem oder allen Energieberatern den Zugriff auf bestimmte Daten freigeben möchte, so wird dies über die Gewährung auf den Zugriff der entsprechenden Private-Keys ermöglicht. Bei Kundeninformationsdatensätzen, Immobiliendatensätzen und Dokumentendatensätzen wird dabei der Zugriff auf den Private-Key des individuellen Datensatzschlüssels gewährt. Bei Messwertdatensätzen wählt der

Kunde den Zeitraum aus, für welchen die Messdaten freigegeben werden sollen. Dabei werden anschließend alle Wochenschlüssel des entsprechenden Schlüsselbandes zugänglich gemacht, wobei es nicht möglich ist, die Daten eines einzelnen Zählers freizugeben, da alle Messwertdatensätze aller Zähler einer Immobilie mit den selben Wochenschlüsseln verschlüsselt sind, da diese vom selben Schlüsselband stammen. D.h. es können immer nur alle Messdaten dieser Immobilie in diesem bestimmten Zeitraum freigegeben werden. Der Grund für dieses Vorgehen ist, dass eine sinnvolle Beratung nur durchführbar ist, wenn ein Berater alle Daten einer Immobilie erhält. Der Freigabezeitraum ist hierbei keinesfalls nur auf die Vergangenheit begrenzt sondern kann auch beliebig in der Zukunft gewählt werden. Dafür werden in der Schlüsselverwaltung einfach die entsprechenden Schlüssel für diesen Zeitraum vorausberechnet und anschließend freigegeben. Bei der Freigabe von Messwerten in einem gewissen Zeitraum, müssen die Zählerdatensätze der entsprechenden Immobilie, welche bis dahin möglicherweise nur mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt sind, auch mit einem der freizugebenden Wochenschlüssel verschlüsselt werden, damit der Berater auch die Zähler-Datensätze einsehen kann.

Die Freigabe von Daten bzw. die Freigabe von Schlüsseln an alle Energieberater funktioniert durch die Freigabe an einen bestimmten Benutzer „all“. Wird ein Schlüssel für diesen Benutzer freigegeben, so hat jeder Berater Zugriff auf diesen Schlüssel. Fragt nun ein Energieberater bei der Schlüsselverwaltung nach einer Liste von Schlüsseln eines Kunden, auf welche er zugreifen darf und dieser hat einen Schlüssel nicht diesem Berater explizit, aber dafür dem Benutzer „all“ freigegeben, so erscheint dieser Schlüssel in der angeforderten Liste, was wiederum bedeuetet, dass der Berater Zugriff auf diesen Schlüssel hat.

4.9.1 Freigabestufen

Einem Kunden stehen folgende Stufen der Sichtbarkeit/Freigabe von Daten für Berater zur Verfügung:

1. Kundenprofil nicht sichtbar: Der Kunde entscheidet, dass er nicht von Beratern gefunden werden kann und schaltet daher sein Profil auf *nicht sichtbar*. Dies bedeutet, dass kein anderer Benutzer irgendwelche Daten dieses Kunden aufrufen oder entschlüsseln kann. Die Sichtbarkeit des Kundenprofils muss über das Frontend (Webportal) implementiert werden.

2. Kundenprofil sichtbar, aber keine Immobilien sichtbar: Der Kunde aktiviert die Sichtbarkeit seines Kundenprofils und kann daher von Beratern gefunden und kontaktiert werden. Der Kunde gibt aber keine Schlüssel für Immobiliendatensätze frei. Daher kann kein Berater Immobilien des Kunden sehen und auch keine Daten des Kunden entschlüsseln.

3. Kundenprofil sichtbar, Immobilie sichtbar, andere Daten gesperrt: Der Kunde hat sowohl die Sichtbarkeit seines Kundenprofils aktiviert als auch den Private-Key eines Immobiliendatensatzes für einen bestimmten Berater oder alle Berater feigegeben. Damit kann ein Berater, welchem der Zugriff auf den Schlüssel

gewährt wurde, erstens den Immobiliendatensatz entschlüsseln und die darin gespeicherten Informationen lesen und zweitens kann der Berater die Struktur der Daten, welche zu dieser Immobilie gespeichert sind, sehen. Dies bedeutet, dass der Berater die Baumstruktur der Daten zu der Immobilie einsehen und somit spezifische Anfragen an den Kunden senden kann. Dies bedeutet, der Berater kann bestimmte Schlüssel vom Kunden anfordern. Diese Anforderungen werden in der Schlüsselverwaltung gespeichert.

4. Kundenprofil sichtbar, Immobilie sichtbar, weitere Daten freigegeben. Wie zuvor ist das Kundenprofil des Kunden und mindestens eine Immobilie des Kunden sichtbar. Zusätzlich gewährt der Kunden einem Berater/mehreren oder allen Beratern den Zugriff auf einzelne Schlüssel, mit welchen sich weitere Daten entschlüsseln lassen. Dabei kann es sich beispielsweise um zu einer Immobilie gespeicherte Dokumente, Kundendokumente, Zählermesswerte eines bestimmten Zeitraums und die zugehörigen Zählerinformationsdatensätze handeln.

4.10 Beschreibung der Systemprozesse

In diesem Abschnitt werden den in Abschnitt 3.2.3 dargestellten Use Cases detaillierte Systemprozesse zugeordnet. Dazu wird zuerst das Use Case Diagramm aus Abschnitt 3.2.3 verfeinert. Danach wird anhand eines Beispiels einem dieser Use Cases ein Systemprozess zugeordnet und dessen Ablauf genau beschrieben. Die übrigen Systemprozesse wurden aufgrund der besseren Lesbarkeit der Arbeit in den Anhang (Abschnitt A.5) ausgelagert.

4.10.1 Use Case Diagramm

Die beiden Use Case Diagramme (Abbildung 4.8 und Abbildung 4.9) zeigen eine Verfeinerung jenes Use Case Diagramms, welches in Abschnitt 3.2.3 vorgestellt wurde. Diese zeigen, welcher Use Case welche Aktionen welcher Komponente nutzt. Aufgrund der Übersichtlichkeit wurden die Use Cases auf zwei Bilder aufgeteilt. Abbildung 4.8 zeigt alle Use Cases zur Dateneinbringung. Abbildung 4.9 zeigt alle anderen Use Cases.

In Abbildung 4.8 ist deutlich zu sehen, dass nur der Kunde und die automatischen Zähler des Kunden Daten in das System einbringen können. Da alle Use Cases zur Dateneinbringung sehr ähnlich ablaufen, nutzen diese auch nahezu die gleichen Aktionen in den verschiedenen Komponenten: Dazu gehört beim Datenspeicher die Aktion *Daten aushändigen*, welche zur Abfrage von Datensätzen und Metadatensätzen dient. Vor dem Speichern eines neuen Datensatzes muss das Kundenlock und eine neue Datensatz-ID angefordert werden. Anschließend werden die Daten und die aktualisierten Metadaten gespeichert und das Kundenlock gelöst. Sollte ein Request von einer automatischen Datenquelle beim Datenspeicher eintreffen, so muss die Signatur des Requests überprüft werden. Dazu fordert der Datenspeicher den entsprechenden Public-Key des Signaturschlüsselpaares der Datenquelle bei der Schlüsselverwaltung an. Anschließend kann die Signatur auf Gültigkeit überprüft werden.

Bei der Schlüsselverwaltung werden für jeden der Use Cases zur Dateneinbringung sowohl Public- als auch Private-Keys abgefragt. Die Public-Keys dienen dabei zum Verschlüsseln des neuen Datensatzes und der aktualisierten Metadatensätze. Beim Use Case *Immobilie anlegen* wird außerdem in der Schlüsselverwaltung ein neues Schlüsselband angelegt. Dieses dient zur Verwaltung der Wochenschlüssel der neuen Immobilie. Im Use Case *Inventarisierung eines Zählers* wird außerdem ein neues Signaturschlüsselband erzeugt und dessen Private-Key ausgehändigt.

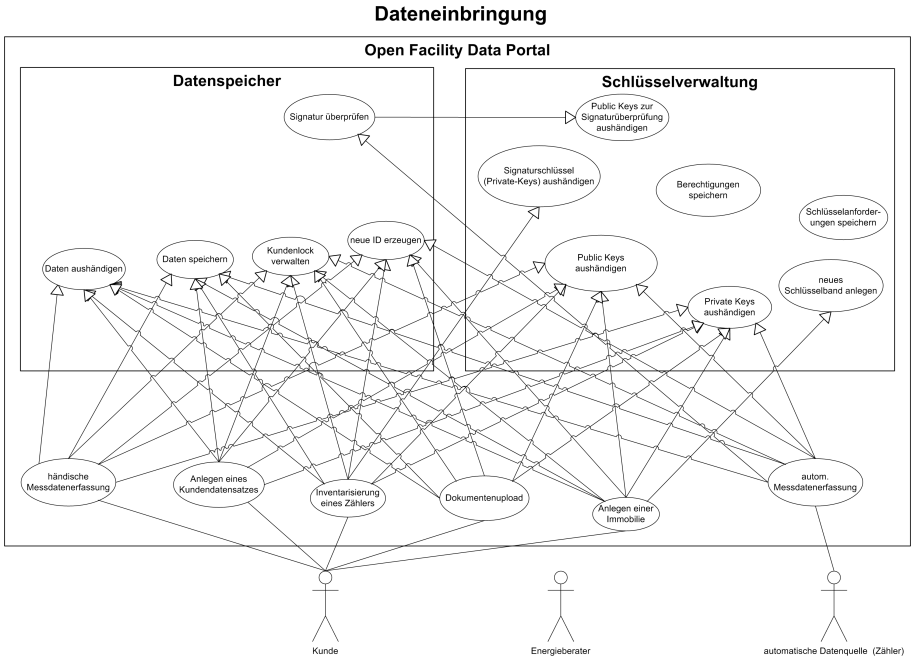


Abbildung 4.8: Use-Case-Diagramm der Use Cases zur Dateneinbringung

Die in Abbildung 4.9 dargestellten Use Cases verwenden meist unterschiedliche Aktionen der Komponenten, auf welche hier kurz eingegangen wird. Der Use Case *Abfrage von Daten* darf sowohl vom Kunden als auch vom Energieberater ausgeführt werden. Dabei werden beim Datenspeicher die gewünschten Daten angefordert. Anschließend werden bei der Schlüsselverwaltung die Private-Keys zur Entschlüsselung dieser Daten angefordert. Die Schlüsselverwaltung überprüft hierbei, ob der Kunde bzw. Energieberater die nötigen Berechtigungen für diesen Schlüssel besitzt. Der Use Case *Editieren von Datenobjekten* darf nur vom Kunden ausgeführt werden. Dabei lädt der Kunde vor dem Editieren den Datensatz mithilfe des Use Cases *Abfrage von Daten*. Beim Speichern des aktualisierten Datensatzes, werden bei der Schlüsselverwaltung die Public-Keys zur Wiederverschlüsselung angefordert. Anschließend wird das Kundenlock beim Datenspeicher angefordert und die Daten gespeichert.

Der Use Case *Anforderung von Daten* wird ausschließlich vom Energieberater

ausgeführt. Dabei lädt dieser die Metadatensätze des Kunden aus dem Datenspeicher und entschlüsselt diese mit den entsprechenden Private-Keys, welche er zuvor bei der Schlüsselverwaltung anfordert. Anschließend wählt der Berater die anzufordernden Datensätze aus und speichert die Anforderung in der Schlüsselverwaltung.

Zur Freigabe von Daten durch den Kunden, lädt dieser die gewünschten Datensätze aus dem Datenspeicher und fordert bei der Schlüsselverwaltung die zugehörigen Schlüssel an. Anschließend wählt der Kunde die freizugebenden Datensätze aus und speichert die Berechtigungen der Schlüsselzugriffe in der Schlüsselverwaltung. Beim Use Case *Freigabe von Messdaten* müssen zusätzlich vor dem Speichern der Berechtigungen die Zählerdatensätze eventuell neu verschlüsselt werden. Dazu werden die entsprechenden Public-Keys bei der Schlüsselverwaltung angefordert. Danach werden die Datensätze neu verschlüsselt und das Kundenlock beim Datenspeicher angefordert. Danach werden die aktualisierten Datensätze gespeichert und das Kundenlock wieder gelöst.

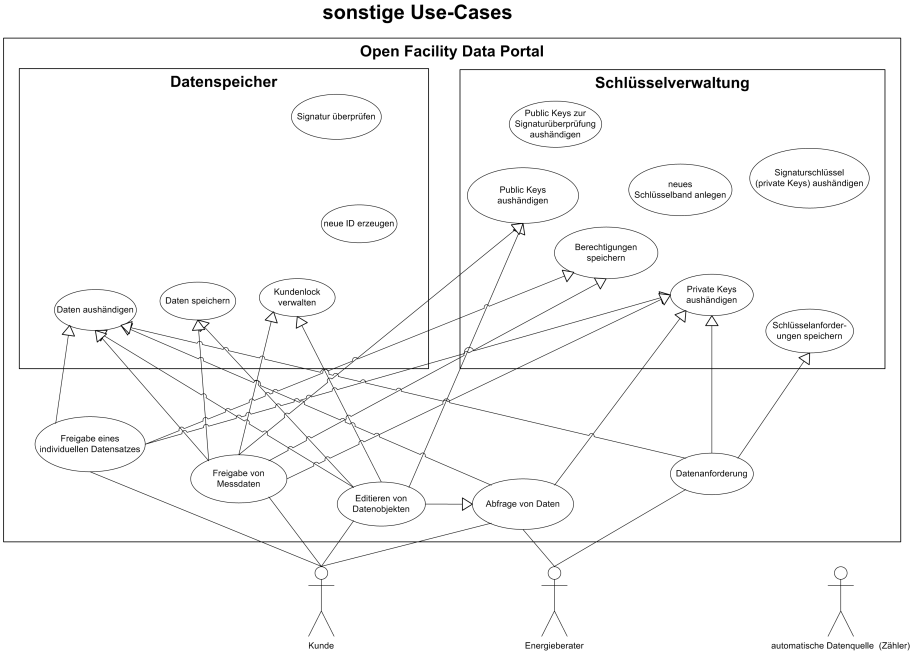


Abbildung 4.9: Use-Case-Diagramm aller anderen Use Cases

4.10.2 Systemprozess zur automatischen Messwerterfassung

Hier soll am Beispiel der automatischen Messwerterfassung durch eine automatische Datenquelle gezeigt werden, welche Abläufe bei diesem Prozess einzuhalten sind bzw. welche Aktionen dabei gesetzt werden müssen, um einen Zählerstand im Datenspeicher verschlüsselt zu speichern. Alle anderen Systemprozesse sind in gleicher

Weise in Abschnitt A.5 im Anhang zu finden. Eine Beschreibung der in der Grafik verwendeten Symbole ist in Abschnitt A.5.2 zu finden.

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar)
- die Datenquelle hat einen neuen Messwert und den zugehörigen Messzeitpunkt
- die Datenquelle ist richtig parametriert

Beschreibung: Nachdem die automatische Datenquelle (z.B. ein automatischer Stromzähler) den Messwert ermittelt hat, wird zuerst überprüft, ob der aktuelle Wochenschlüssel schon im Zähler vorhanden ist. Hier wird davon ausgegangen, dass der letzte Wochenschlüssel abgelaufen ist und daher ein neuer angefordert werden muss. Hierzu erstellt die Datenquelle einen Request zur Anforderung des Public-Key des aktuellen Wochenschlüssels des parametrierten Schlüsselbandes und signiert diesen mit dem Private-Key des Signaturschlüsselpaares, welcher ebenfalls im Zähler parametriert ist. Anschließend wird der Request an die Schlüsselverwaltung geschickt, welche die Signatur des Requests mit dem entsprechenden Public-Key überprüft und anschließend den geforderten Public-Key des Wochenschlüssels zurückliefert. Danach erzeugt die automatische Datenquelle einen neuen Messwertdatensatz und verschlüsselt diesen symmetrisch mit einem neu generierten Session-Key. Danach wird dieser Session-Key mit dem zuvor angeforderten Public-Key des Wochenschlüssels und dem Public-Key des Backup-Schlüsselpaares, welcher ebenfalls in der Datenquelle parametriert ist, asymmetrisch verschlüsselt. Danach wird mit einem signierten Request beim Datenspeicher eine neue Datensatz-ID für den neuen Messwertdatensatz angefordert. Der Datenspeicher überprüft die Signatur (siehe Systemprozess 5.1 - Abschnitt A.5.8) und liefert anschließend eine neue Datensatz-ID.

Danach fordert die Datenquelle das Lock des Kunden beim entsprechenden Datenspeicher an. Nachdem dieser die Signatur des Requests überprüft hat, wird bei Erhalt des Locks der Lockstring an die Datenquelle übermittelt. Anschließend fordert die Datenquelle den Metadata-Datensatz des Zählers beim Datenspeicher an (siehe Systemprozess 5.4 - Abschnitt A.5.8). Anschließend überprüft die Datenquelle, ob für das aktuelle Jahr und die aktuelle Kalenderwoche schon Metadata-Datensätze existieren. Sollte dies der Fall sein, so werden diese beim Datenspeicher angefordert (siehe Systemprozess 5.4 - Abschnitt A.5.8). Andernfalls werden diese angelegt und dazu auch neue Datensatz-IDs beim Datenspeicher angefordert (wie oben schon für den Messwertdatensatz). Anschließend wird der neue Messwertdatensatz in den Metadata-Datensatz der entsprechenden Kalenderwoche eingetragen. Anschließend werden alle neuen oder geänderten Metadata-Datensätze mit einem neuen Session-Key symmetrisch verschlüsselt und der Session-Key mithilfe des Public-Keys des Metadata-Schlüsselpaares (in der Datenquelle parametriert) und des Public-Keys des

Systemprozess: 1.1 Automatische Messdatenerfassung (online)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar); die Datenquelle hat einen neuen Messwert und den zugehörigen Messzeitpunkt; die Datenquelle ist richtig parametriert
Kurzbeschreibung: Die Datenquelle fordert bei der Schlüsselverwaltung einen neuen Wochenschlüssel an. Danach werden die Daten mit dem aktuellen Wochenschlüssel und mit dem Public Key des Backup-Schlüsselpaares verschlüsselt und beim Datenspeicher eine neue Datensatz-ID angefordert. Nachdem die Datenquelle das Kunden-Lock angefordert hat, liest diese den Metadatensatz des Zählers und legt neue Metadatensätze an bzw. aktualisiert diese (trägt den neuen Messwert ein). Danach werden die Daten im Datenspeicher gesichert und das Lock gelöst.

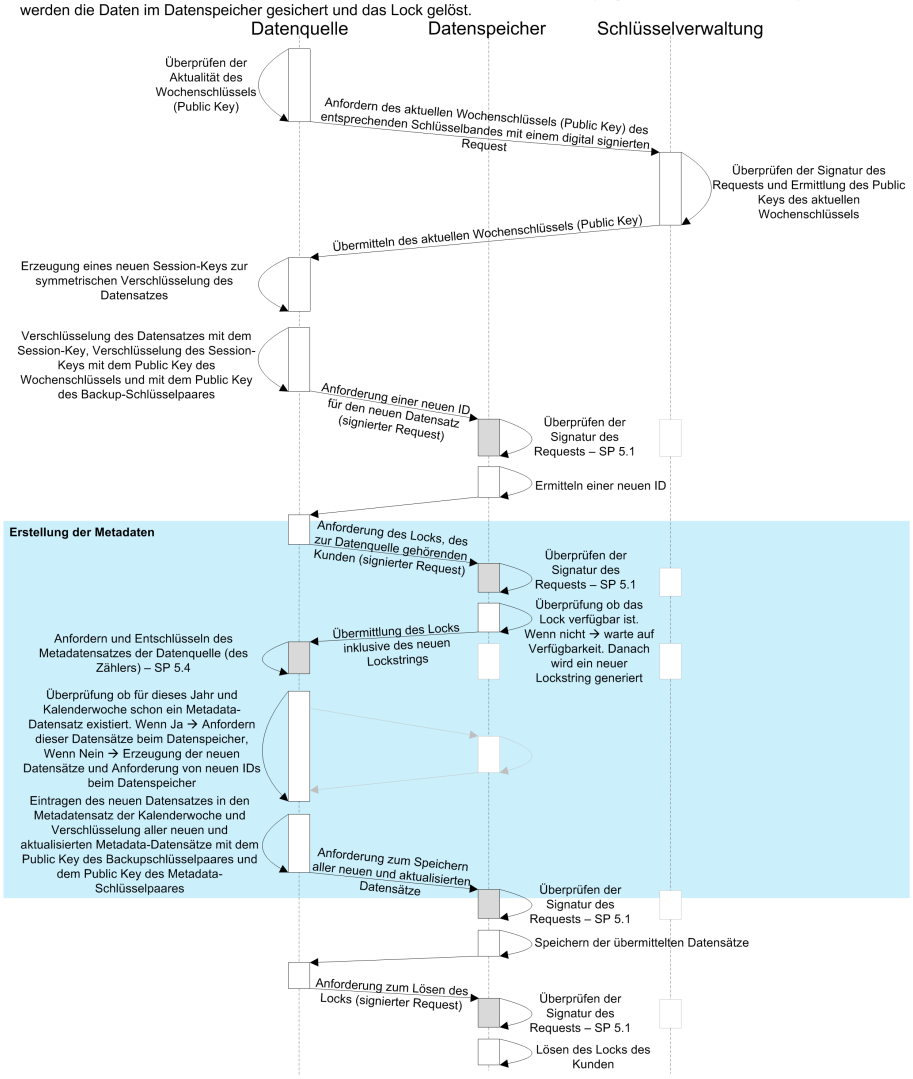


Abbildung 4.10: Systemprozess 1.1 Automatische Messdatenerfassung (online)

Backup-Schlüsselpaares verschlüsselt. Anschließend werden alle neuen oder aktualisierten Datensätze an den Datenspeicher geschickt. Dieser speichert die Daten in der Datenbank, nachdem die Signatur des Requests überprüft wurde. Anschließend fordert die Datenquelle die Lösung des Kundenlocks beim Datenspeicher an. Dieser überprüft wiederum die Signatur und löst anschließend das Lock des Kunden.

4.11 Beschreibung der Interfaces

Einen weiteren wichtigen Designschritt stellt die Definition geeigneter Schnittstellen zwischen den Komponenten dar. Nachdem im vorigen Abschnitt die Systemprozesse beschrieben wurden, in welchen sehr detailliert auf die Clientseite eingegangen wurde, werden bei der Schnittstellendefinition die serverseitigen Funktionen und deren Aufgaben genau beschrieben. Da es sich bei der Beschreibung der Schnittstellen um ein sehr umfangreiches Kapitel handelt, welches vor allem bei der Implementierung des Systems relevant ist, wurde dieser Abschnitt aufgrund der besseren Lesbarkeit ebenfalls in den Anhang (Abschnitt A.6) ausgelagert. Dort werden alle Funktionen des Datenspeichers und der Schlüsselverwaltung hinsichtlich der verschiedenen Clients genau beschrieben. Die Beschreibung umfasst folgende Schnittstellen:

- Schnittstellen zur Schlüsselverwaltung
  - Schnittstelle zwischen Kunden- bzw. Energieberaterclient und der Schlüsselverwaltung (Abschnitt A.6.1)
  - Schnittstelle zwischen automatischer Datenquelle und Schlüsselverwaltung (Abschnitt A.6.2)
  - Schnittstelle zwischen Datenspeicher und Schlüsselverwaltung (Abschnitt A.6.3)
- Schnittstellen zum Datenspeicher
  - Schnittstelle zwischen Kunden- bzw. Energieberaterclient und dem Datenspeicher (Abschnitt A.6.4)
  - Schnittstelle zwischen automatischer Datenquelle und Datenspeicher (Abschnitt A.6.5)

Neben der detaillierten Beschreibung der Schnittstelle findet man im Anhang auch eine Interfacedefinition in Form von WSDL[2]-Dateien, welche für die Implementierung wichtig ist.
Als nächstes soll noch gezeigt werden, wie mit Änderungen an der Schnittstelle umgegangen werden kann, ohne dass es zu Inkompatibilitäten kommt.

[2] Web Service Description Language

4.11.1 Versionierung der Schnittstelle

Dieser Abschnitt soll zeigen, wie eine geordnete Versionierung der Schnittstelle aussehen kann. Dies bedeutet, dass eine Vorgangsweise vorgestellt wird, wie mit Veränderungen an der Schnittstellendefinition umgegangen werden kann, ohne dass es auf der Clientseite oder auf der Serverseite zu Problemen oder Inkonsistenzen kommt.

Grundsätzlich muss bei der Versionierung von Web-Service-Schnittstellen zwischen zwei Arten von Änderungen unterschieden werden. Zum einen gibt es Änderungen an der Schnittstelle welche, Clients, die die Änderung noch nicht kennen, unberührt lassen (kompatible Änderungen). Zum anderen gibt es inkompatible Änderungen an der Schnittstelle, welche ein Update des Clients zwingend notwendig machen (siehe [23] und [6]).

Zu den kompatiblen Änderungen gehört beispielsweise das Hinzufügen von neuen Funktionen oder neuen Typdefinitionen zu einem Web-Service. Clients welche von der Veränderung nichts mitbekommen haben, funktionieren weiterhin problemlos, da sie von der Veränderung unberührt bleiben. Ähnlich verhält es sich beim Hinzufügen von Attributen zu Typen, welche das Service verwendet. Dies gilt allerdings nur, wenn diese Typen als erweiterbar gekennzeichnet wurden. Dies bedeutet, dass wann immer ein Client ein Objekt eines Typs empfängt, welches mehr Attribute hat, als der Client erwarten würde, er die zusätzlichen Attribute einfach ignoriert. Sollte der Typ nicht als erweiterbar gekennzeichnet sein, handelt es ich um eine inkompatible Änderung.

Alle anderen Änderungen sind inkompatible Änderungen. Dazu gehören das Löschen von Funktionen, das Löschen von Typdefinitionen, das Ändern von Typdefinitionen bzw. das Ändern von Datentypen und das Umbenennen von Funktionen oder Typen. Inkompatible Änderungen erfordern es immer, ein neues Service zu erstellen. Dies bedeutet, dass ein neues seperates WSDL-Dokument veröffentlicht werden muss, welches einen neuen Namespace verwendet. Dieser Namespace muss eindeutig der neuen Version zuzuordnen sein. Dazu kann entweder eine eindeutige Versionsnummer (z.B. `http://myexample.com/v2_4/mytype.xsd`) oder das Datum der Veröffentlichung (z.B. `http://myexample.com/2010/01/15/mytype.xsd`) im Namespace verwendet werden.

Da der Namespace in jedem SOAP-Request automatisch mitgesendet wird, kann mithilfe dieses eindeutigen Namespaces serverseitig genau festgestellt werden, auf welche Version sich der Client bezieht und der Server kann die Requests an das richtige Service weiterleiten.

Neben der Verwendung von eindeutigen Namespaces wäre es auch möglich, für jede Version der Schnittstelle eine eigene URL zu verwenden.

Sollte eine Service-Version nicht mehr vom Server unterstützt werden, so ist dies dem Client durch eine geeignete Fehlernachricht mitzuteilen. Optimal wäre, die Clients regelmäßig über Updates zu informieren, damit diese rechtzeitig die notwendigen Maßnahmen ergreifen können, um mit dem Service kompatibel zu bleiben.

4.12 Historisierung des Datenspeichers

Zum Abschluss des Designkonzepts soll dieser Abschnitt nun ein Konzept zur historisierten Speicherung der Daten der Kunden vorstellen. Da die Historisierung und Ver-

sionierung der Daten, welche in einer Datenbank gespeichert sind, ein immer wichtigeres Thema wird, wird in diesem Abschnitt darauf eingegangen, wie eine Umsetzung, welche eine historisierte Datenhaltung ausweist, aussehen könnte. Die Historisierung der Daten kann bei dem vorliegenden System eine große Relevanz erlangen: Wie in Abschnitt 6.1 erläutert, spielt die korrekte Implementierung der Clients eine essenzielle Rolle bei der Konsistenzhaltung der Daten. Angenommen ein Client würde, beispielsweise aufgrund eines Updates, welches einen Fehler enthält, Datensätze eines Kunden so zerstören, dass sie nicht mehr oder nur noch teilweise zu gebrauchen sind. In solch einem Fall wäre es vorteilhaft, wenn man auf den Datenbestand des Kunden vor dem fehlerhaften Update zurückgreifen könnte.

Es sei allerdings vorweggenommen, dass die historisierte Speicherung der Daten auch zu einem erhöhten Speicherbedarf führt. Dies wird vor allem durch die Verschlüsselung der Metadaten besonders deutlich, da jedes mal, wenn ein Datensatz hinzukommt, ein Update auf einen Metadatensatz durchgeführt werden muss und somit dieser Metadatensatz durch die Historisierung nochmals gespeichert wird. In Abschnitt 6.4 wird auf diesen Sachverhalt genauer eingegangen.

Nachdem nun auf die Vor- und Nachteile der historisierten Datenspeicherung eingegangen wurde, wird nun beschrieben, wie diese im vorliegenden System umgesetzt werden kann. Dabei wird vor allem auf die Komponente des Datenspeichers eingegangen. Bei der Schlüsselverwaltung kann die Vorgangsweise analog angewendet werden. Die Implementierung der Historisierung der Daten im Datenspeicher kann prizipiell entweder im Web-Service des Datenspeichers oder in der Datenbank durch entsprechende Trigger und Views umgesetzt werden (siehe hierzu [17], [8]). Die zweite Methode hätte den Vorteil, dass die Historisierung transparent zum Web-Service durchgeführt werden kann. Aus diesem Grund wird hier vor allem auf die Implementierung mittels Trigger und Views eingegangen (vgl. [17]).

Für das Datenbankschema ist die Wahl der Implementierung egal. Das bedeutet, dass am Schema in beiden Arten die gleichen Änderungen durchgeführt werden müssen: Jede Tabelle erhält zusätzlich zur ID, welche den Datensatz bisher eindeutig identifiziert hat, eine Versionsnummer, welche auch ein *Primary Key* Attribut wird. Der neue *Primary Key* besteht also aus der ID gemeinsam mit der Versionsnummer. Als Typ dieser Versionsnummer wird ein Integertyp gewählt. Zusätzlich zur Versionsnummer muss in jeder Tabelle eine Spalte existieren, welche den Zeitpunkt des Anlegens der aktuellen Version des Datensatzes wiedergibt. Diese Spalte ist im aktuellen Schema schon vorhanden (Spalte *lastWriteTime*).

Im aktuellen Konzept ist ein Löschen von Datensätzen nicht vorgesehen. Jedoch sollte bei der Historisierung der Daten Rücksicht darauf genommen werden: Anstatt Datensätze wirklich zu löschen, sollten diese nur als gelöscht markiert werden. Dies kann über eine Spalte, welche ein Delete-Flag darstellt, implementiert werden. Als Typ ist ein Boolean-Typ ausreichend. Zusätzlich dazu wird auch noch der Zeitpunkt des Löschvorganges festgehalten. Dies geschieht ebenfalls durch Einfügen einer neuen Spalte. Diese ist immer *NULL*, solange der Datensatz nicht gelöscht wurde. Sollte der Datensatz als gelöscht markiert worden sein, wird in diese Spalte der Zeitpunkt des Löschvorganges eingetragen.

4.12.1 Hinzufügen eines Datensatzes

Für das Hinzufügen eines neuen Datensatzes wird ein *INSTEAD OF*-Trigger angewendet, welcher anstatt des normalen *Insert*s ein *Insert* mit der Versionsnummer *0* und dem aktuellen Zeitstempel durchführt. Das Delete-Flag wird auf „nicht gelöscht“ gesetzt.

4.12.2 Update eines Datensatzes

Auch für das *Update* wird ein *INSTEAD OF*-Trigger angewendet, welcher anstatt des normalen *Update*s ein *Insert* mit einer um 1 erhöhten Versionsnummer und dem aktuellen Zeitstempel durchführt. Das Delete-Flag wird wiederum auf „nicht gelöscht“ gesetzt. Die ID des Datensatzes bleibt die gleiche. Dies wird deshalb von der Datenbank akzeptiert, da die Versionsnummer auch ein *Primary Key* Attribut darstellt und die ID gemeinsam mit der inkrementierten Versionsnummer den Datensatz eindeutig identifiziert. Das *Update* entspricht also einem *Add* einer neuen Version des Datensatzes, wobei der alte Datensatz unverändert bleibt. Vor dem *Update* muss außerdem geprüft werden, ob der Datensatz nicht schon als gelöscht markiert wurde. Sollte dies der Fall sein, so wird das Update nicht durchgeführt.

4.12.3 Löschen eines Datensatzes

Für das Löschen eines Datensatzes wird ebenfalls ein *INSTEAD OF*-Trigger angewendet, welcher ein *Update* auf den Datensatz mit der aktuellsten Versionsnummer durchführt. Bei diesem Update, wird das Delete-Flag auf den Wert „gelöscht“ gesetzt und der Zeitpunkt des Löschvorganges in der entsprechenden Spalte vermerkt.

4.12.4 Lesen eines Datensatzes

Zum Lesen eines Datensatzes kann eine View erstellt werden, welche von dem geforderten Datensatz die aktuellste Version liefert, sofern der Datensatz noch nicht als gelöscht markiert wurde.

4.12.5 Herstellen des Datenbestandes eines Kunden aus der Vergangenheit

Möchte man nun den Datenbestand eines Kunden zu einem gewissen Zeitpunkt in der Vergangenheit wiederherstellen, so können einfach alle Datensätze, welche einen Versionszeitstempel kleiner als der gewünschte Zeitpunkt haben, aufgehoben werden. Alle neueren Datensätze können nach einem Backup (welches bei solchen Vorgängen immer zu empfehlen ist) verworfen werden.

Sollte der gewünschte Zeitpunkt zwischen dem Versionszeitpunkt und dem Delete-Zeitpunkt eines Datensatzes liegen, so muss dieser Datensatz wieder als „nicht gelöscht“ markiert werden.

4.12.6 Aufräumen der Historie

Da sich im Laufe der Zeit, vor allem bei den Metadaten, eine große Anzahl verschiedener Versionen eines Datensatzes anhäufen, welche teilweise schon recht alt sein können, ist es sinnvoll, zu alte Versionen zu verwerfen, um wieder Speicherplatz freizugeben. Beispielsweise könnte man die Historie der Daten für das letzte Jahr behalten und alle älteren Daten löschen. Jedoch ist bei diesem Schritt mit äußerster Vorsicht vorzugehen und ein vorhergehendes Backup wird angeraten.

Die Vorgangsweise sieht folgendermaßen aus: Man wählt einen Zeitpunkt in der Vergangenheit, bis zu welchem die Historie der Daten gelöscht werden soll. Dieser Zeitpunkt wird im weiteren als *neuer Historienbeginn* bezeichnet. Nun können alle Versionen eines Datensatzes, welche einen Versionszeitpunkt vor der letzten Version vor dem *neuer Historienbeginn* haben, gelöscht werden (siehe Abbildung 4.11).

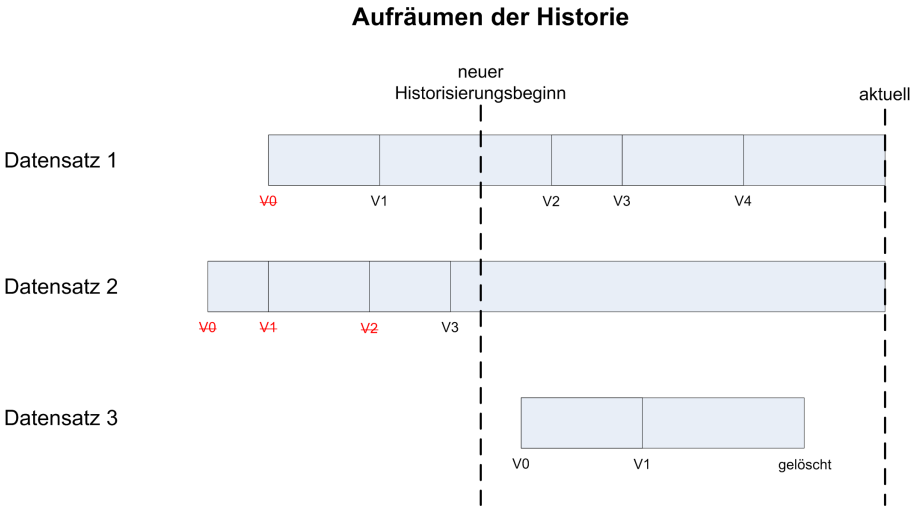


Abbildung 4.11: Aufräumen der Historie - die roten (durchgestrichenen) Versionen werden gelöscht; alle anderen Versionen bleiben erhalten

Diese Vorgangsweise sichert, dass von allen Datensätzen die aktuelle Version und die Historie bis zum gewählten *neuen Historisierungsbeginn* erhalten bleibt.

4.13 Zusammenfassung

In diesem Kapitel wurde das Designkonzept des *Open Facility Data Portals* vorgestellt. Dazu wurde eingangs beschrieben, wie die automatische Messdatenerfassung mithilfe von automatischen Zählern funktioniert. Danach wurde der Aufbau der Daten eines Kunden vorgestellt. Anschließend wurde auf die Verschlüsselung der Datensätze und auf die dazu verwendeten Schlüssel eingegangen. Danach wurde beschrieben, wie Metadaten, welche zur Identifikation und zum Wiederauffinden der Datensätze notwendig sind, verschlüsselt im System gespeichert werden können. Dies ist notwendig,

damit der Datenspeicheranbieter die Struktur und Art der Daten nicht erkennen kann. Diese verschlüsselte Speicherung der Metadaten machte es notwendig, einen Lock-Mechanismus zu entwickeln, welcher es einem Client ermöglicht, für eine gewisse Zeit exklusiven Schreibzugriff auf den Datenspeicher zu erlangen. Danach wurden die schon in Kapitel 3 vorgestellten Komponenten näher beschrieben und deren Aufgabenbereiche genau definiert. Anschließend wurde der Vorgang der Weitergabe von Daten durch den Kunden an einen Energieberater detailliert dargestellt. Hiernach wurden die in der Analyse identifizierten Use Cases auf Systemprozesse abgebildet, deren Ablauf genau beschrieben wurde. Anschließend wurden die Schnittstellen zwischen den einzelnen Komponenten definiert. Abgeschlossen wurde das Kapitel durch die Vorstellung eines Konzepts zur historisierten Speicherung der Daten eines Kunden.

KAPITEL 5

Demo-Applikation

Zur Verifikation der Vollständigkeit und Korrektheit des in Kapitel 4 beschriebenen Designkonzepts, wurde eine Demo-Applikation erstellt, welche alle Systemprozesse und Schnittstellenfunktionalitäten (siehe Abschnitt 4.10 bzw. Abschnitt 4.11) implementiert. Die in Abschnitt 4.12 vorgestellte historisierte Speicherung der Daten wurde nicht implementiert. Als Entwicklungsumgebung für die Demo-Applikation wurde *Visual Studio 2008* verwendet. Eine kurze Anleitung zur Inbetriebnahme der Demo-Applikation befindet sich in Abschnitt A.8.

Die Demo-Applikation besteht aus folgenden 6 Komponenten:

- Schlüsselverwaltung
- Datenspeicher 1
- Datenspeicher 2[1]
- Kundenclient
- Beraterclient
- Automatischer Zähler

Die ersten drei Komponenten stellen dabei die Server-Komponenten dar. Die restlichen Komponenten bilden die Clients.

In den folgenden Abschnitten werden die einzelnen Komponenten nun näher vorgestellt. Bei den Server-Komponenten wird dabei auch auf die Speicherung der Daten in einer Datenbank eingegangen. Danach wird anhand eines Systemprozesses die Funktion der Demo-Applikation illustriert. Zuerst wird allerdings noch ein kurzer Blick auf

[1] *Datenspeicher 1* und *Datespeicher 2* sind idente Instanzen des Datenspeichers

die Entwicklungsumgebung geworfen.

Wie schon erwähnt, wurde zur Entwicklung *Visual Studio 2008*, verwendet wobei die Demo-Applikation in *C#* implementiert wurde. Die beschriebenen Schnittstellen zur Schlüsselverwaltung und zum Datenspeicher wurden mittels *WCF*[2] realisiert (siehe [5]).

WCF ist eine Programmierschnittstelle des *.NET-Frameworks* (ab der Version 3.0), welche es erlaubt, serviceorientierte verteilte Applikationen zu erstellen. Dabei spielt die Art des Nachrichtenkanals (das sogenannte Binding) bei der Programmierung des Services und der Service-Schnittstelle keine Rolle. Das Binding kann später durch einfache Konfiguration festgelegt werden, ohne dass der Source-Code dabei verändert oder neu kompiliert werden muss. *WCF* unterstützt eine Vielzahl von Übertragungsprotokollen (wie z.B. HTTP[3], TCP, Named Pipes, UDP, MSMQ,...), Serialisierungsformaten (wie z.B. SOAP[4], MTOM, JSON,...) sowie WS-*-Standards zur Authentifizierung, Verschlüsselung, Autorisierung, etc. [3], [7]. Unter anderem besteht die Möglichkeit, dass ein Service gleichzeitig über mehrere unterschiedlich konfigurierte Kommunikationskanäle angeboten wird. So könnte ein Service innerhalb eines Netzwerks über eine TCP-Verbindung mit binärer Serialisierung und von außen über das Internet über eine interoperable HTTP/SOAP-Schnittstelle erreicht werden.

Eine weitere Eigenschaft von WCF ist es, dass Services auf die verschiedensten Weisen gehostet werden können. So kann ein Service beispielsweise über ein ganz normales .NET-Programm (Konsolenapplikation oder Windows-Forms-Anwendung) oder auch über Microsoft IIS[5] gehostet werden.

5.1 Komponente *Schlüsselverwaltung*

Die Schlüsselverwaltung dient zum Anlegen, Speichern und Freigeben von kryptographischen Schlüsseln eines Kunden. Der implementierte Schlüsselverwaltungsserver bietet insgesamt drei Schnittstellen (also drei verschiedene Service Contracts), wobei jede für einen anderen Typ von Client ausgelegt ist. Dabei gibt es eine Schnittstelle für die Kommunikation mit normalen Clients, eine für die Kommunikation mit automatischen Datenquellen und eine für die Kommunikation mit dem Datenspeicher. Die Kommunikation mit dem Datenspeicher dient dazu, dass der Datenspeicher Public-Keys von Signaturschlüsselpaaren bei der Schlüsselverwaltung abfragen kann und somit die Signatur eines Requests von einer automatischen Datenquelle überprüfen kann. Als Host für die Schlüsselverwaltungsservices wurde eine Konsolenapplikation verwendet.

Dadurch, dass für jede Art von Client eine eigene Schnittstelle implementiert wurde, kann jede Schnittstelle sehr einfach auf die konkreten Bedürfnisse des Clients

[2] Windows Communication Foundation
[3] Hypertext Transfer Protocol
[4] Simple Object Access Protocol
[5] Internet Information Services

abgestimmt werden. So könnte beispielsweise für die Kommunikation mit normalen Clients ein anderes Binding verwendet werden, als für die Kommunikation mit automatischen Datenquellen.
Die folgende Auflistung zeigt die Adressen, Service-Endpoints und Bindings der verschiedenen Schnittstellen:

Schnittstelle zwischen normalen Clients und Schlüsselverwaltung:

- **Adresse:** `http:\\localhost:8010`
- **Endpoint:** KeyManagement
- **Binding:** wsHttpBinding
- **Beschreibung:** Diese Schnittstelle dient zur Kommunikation mit normalen Clients (Kundenclient oder Beraterclient). Sie bietet alle in Abschnitt A.6.1 beschriebenen Funktionen. Das dazugehörige WSDL-File befindet sich im Anhang. (A.7.1)

Schnittstelle zwischen automatischer Datenquelle und Schlüsselverwaltung:

- **Adresse:** `http:\\localhost:8013`
- **Endpoint:** KeyManagementForAutomaticMeter
- **Binding:** wsHttpBinding
- **Beschreibung:** Diese Schnittstelle dient zur Kommunikation mit automatischen Datenquellen. Die automatische Datenquelle kann über diese Schnittstelle Public-Keys von Wochenschlüsselpaaren des zugehörigen Kunden abfragen. Alle Requests, welche über diese Schnittstelle empfangen werden, müssen digital signiert sein. Der Server überprüft anhand der Signatur des Requests mithilfe des gespeicherten Public-Keys des Signaturschlüsselpaares die Gültigkeit der Signatur. Diese Schnittstelle bietet alle in Abschnitt A.6.2 beschriebenen Funktionen. Das dazugehörige WSDL-File und das XML-Schema zur Serialisierung der Requests befinden sich im Anhang (A.7.2).

Schnittstelle zwischen Datenspeicher und Schlüsselverwaltung:

- **Adresse:** `http:\\localhost:8016`
- **Endpoint:** KeyManagementForDataStorage
- **Binding:** wsHttpBinding
- **Beschreibung:** Diese Schnittstelle dient zur Kommunikation mit den Datenspeichern, welche Public-Keys von Signaturschlüsselpaaren über diese Schnittstelle abfragen können. Diese Schnittstelle bietet alle in Abschnitt A.6.3 beschriebenen Funktionen. Das dazugehörige WSDL-File befindet sich im Anhang (A.7.3).

Prinzipiell ist zu sagen, dass bei der Implementierung der Schlüsselverwaltung darauf geachtet wurde, dass alle Operationen so einfach und leicht verständlich wie möglich gehalten wurden, da es sich hierbei um eine sehr kritische Komponente im Gesamtsystem handelt. Eine einfache Implementierung beherbergt einerseits weniger potentielle Fehlerquellen und erleichtert darüber hinaus die Fehlersuche. Außerdem kann dadurch die Verifikation der korrekten Funktionsweise sehr viel einfacher durchgeführt werden.

Prinzipiell muss die Schlüsselverwaltung dafür Sorge tragen, dass nur der Kunde selbst (oder die automatischen Datenquellen dieses Kunden) Zugriff auf die Public-Keys des Kunden hat. Außerdem dürfen nur jene Berater, welche auch vom Kunden dazu berechtigt wurden, auf ausgewählte Private-Keys des Kunden zugreifen. Auch die Zugriffsberechtigung auf Private-Keys eines Kunden darf nur vom Schlüsselbesitzer (Kunde) selbst gewährt werden. Der Kunde selbst darf natürlich auf alle eigenen Private-Keys zugreifen.

5.1.1 Speicherung der Daten der Schlüsselverwaltung

In diesem Abschnitt wird darauf eingegangen, wie die Daten der Schlüsselverwaltung und somit die Schlüssel der Kunden in einer SQL-Datenbank gespeichert werden. Bei der Demo-Applikation wurde eine *Microsoft SQL-Express* Datenbank verwendet. Abbildung 5.1 zeigt das verwendete Datenbankschema.
Die Tabelle *user* speichert die Daten eines Benutzers. Momentan wird lediglich der Benutzername (Spalte *userName*) des Benutzers gespeichert.

Die Tabelle *key* bildet eine der wichtigsten Tabellen der Schlüsselverwaltung. In ihr werden die Schlüsselpaare eines Kunden gespeichert. Dabei werden der Public-Key und der Private-Key und die zugehörigen deskriptiven Daten eines Schlüssels gespeichert. Zu diesen Daten gehören das Jahr und die Kalenderwoche (Spalten *year* bzw. *weekInYear*), die Art des Schlüsselpaares bzw. die Art des Datensatzes, für welchen dieses Schlüsselpaar vorgesehen ist (Spalte *Purpose*). Außerdem verweist diese Tabelle auf jenen Datenspeicher, in welchem der Datensatz, welcher mit dem Public-Key dieses Schlüsselpaares verschlüsselt wurde, gespeichert ist. Wenn es sich bei dem Schlüssel um einen Wochenschlüssel eines Schlüsselbandes handelt, wird zusätzlich ein Verweis auf dieses Schlüsselband gespeichert. Zusätzlich zum Schlüssel wird außerdem der Algorithmus gespeichert, für welchen das Schlüsselpaar erzeugt wurde (Spalte *algorithm*). In der momentanen Implementierung werden lediglich RSA-Keys unterstützt.

Die Tabelle *keyband* dient zum Speichern von Schlüsselbändern eines Kunden. Wenn ein Kunde ein neues Schlüsselband anlegt, wird in dieser Tabelle ein neuer Eintrag erzeugt, welcher unter anderem den Datenspeicher, für welchen das neuen Schlüsselband vorgesehen ist und den Algorithmus der Schlüssel dieses Schlüsselbandes beinhaltet.

Die Tabelle *dataStorage* speichert die Namen und URLs aller bekannten Datenspeicher.

In der Tabelle *sharingRequest* werden die Schlüsselanfragen eines Beraters an einen Kunden gespeichert. Den Primary-Key dieser Tabelle bilden dabei die *userID*

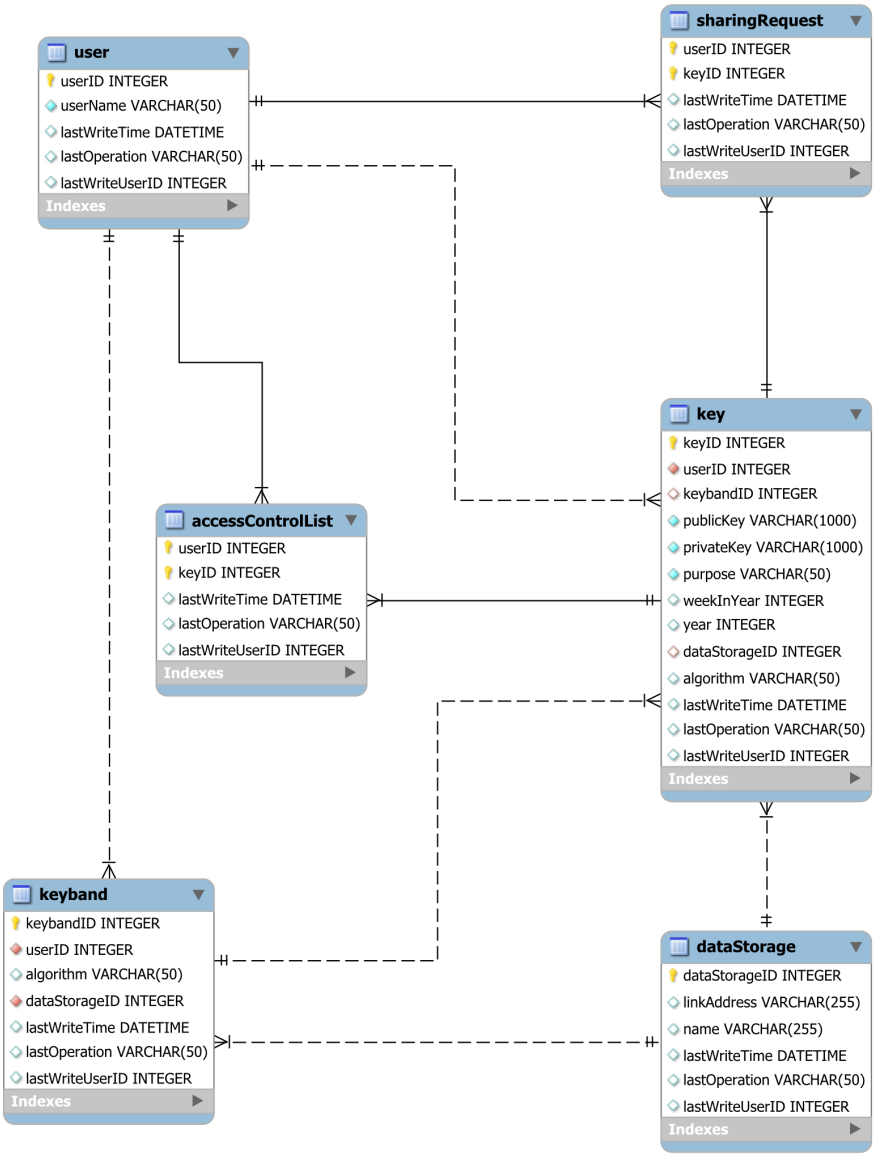


Abbildung 5.1: Datenbankschema der Schlüsselverwaltung

des Beraters, welcher den Schlüssel anfordert und die Schlüssel-ID (*keyID*) des angeforderten Schlüssels.

Die Tabelle *accessControlList* dient schließlich zur Speicherung der Zugriffrechte eines Beraters auf ausgewählte Schlüssel eines Kunden. Der Primary-Key setzt sich auch hier wieder aus der *userID* des Beraters und der *keyID* des Schlüsselpaares, auf wessen Private-Key der Berater Zugriff hat, zusammen.

In jeder Tabelle werden zusätzlich der Zeitstempel des letzten Schreibzugriffs (Spalte *lastWriteTime*), die Art des Schreibzugriffs (Spalte *lastOperation*) und die Benutzer-ID des Benutzers, welcher den Schreibzugriff ausgeführt hat (Spalte *lastWriteUserID*), gespeichert. Dies dient zur einfacheren Ursachenforschung bei eventuell auftretenden Problemen.

5.2 Komponente *Datenspeicher 1* und Komponente *Datenspeicher 2*

Die Datenspeicher dienen zum Ablegen und Speichern der Datensätzes eines Kunden. Ein Kunde kann sein Datenaufkommen in nur einem Datenspeicher ablegen oder dieses auf mehrere Datenspeicher aufteilen. In der Demo-Applikation wurden zwei Datenspeicher implementiert, wobei zwar beide das selbe Service benutzen, welches aber von zwei verschiedenen Hosts angeboten wird. Außerdem benutzt jede Instanz eine eigene Datenbank. Somit kann man diese beiden Datenspeicher als völlig unabhängig voneinander betrachten. Jeder Datenspeicher implememtiert zwei Schnittstellen (also zwei verschiedene Service Contracts). Eine für die Kommunikation mit normalen Clients (Kundenclient bzw. Beraterclient) und eine für die Kommunikation mit automatischen Datenquellen.

Wie zuvor bei der Schlüsselverwaltung, wurde aufgrund der individuellen Bedürfnisse der Clients für jede Art von Client eine eigene Schnittstelle implementiert.
Die folgende Auflistung zeigt die Adressen, Service-Endpoints und Bindings der verschiedenen Schnittstellen:

Schnittstelle zwischen normalen Clients und Datenspeicher:

- **Adresse (Datenspeicher 1):** `http:\\localhost:8011`
- **Adresse (Datenspeicher 2):** `http:\\localhost:8012`
- **Endpoint:** DataStorage
- **Binding:** wsDualHttpBinding
- **Beschreibung:** Diese Schnittstelle dient zur Kommunikation mit normalen Clients (Kundenclient oder Beraterclient). Sie bietet alle in Abschnitt A.6.4 beschriebenen Funktionen. Das dazugehörige WSDL-File befindet sich im Anhang (A.7.4).
  Das *wsDualHttpBinding* könnte bei nicht WCF-Clients Probleme verursachen, da dieses Binding nicht komplett interoperabel ist. Notwendig wurde dieses

Binding durch das Callback des Lock-Mechanismus (siehe Abschnitt 4.6). Sollten Probleme mit diesem Binding, auftreten so könnte alternativ ein zwar weniger effizienter, aber dafür interoperabler Polling-Mechanismus eingeführt werden.

Schnittstelle zwischen automatischer Datenquelle und Schlüsselverwaltung:

- **Adresse (Datenspeicher 1):** `http:\\localhost:8014`
- **Adresse (Datenspeicher 2):** `http:\\localhost:8015`
- **Endpoint:** DataStorageForAutomaticMeter
- **Binding:** wsDualHttpBinding
- **Beschreibung:** Diese Schnittstelle dient zur Kommunikation mit automatischen Datenquellen. Alle Requests, welche über diese Schnittstelle empfangen werden, müssen digital signiert sein. Der Server überprüft anhand der Signatur des Requests und mithilfe des entsprechenden Public-Keys des Signaturschlüsselpaares, welches bei der Schlüsselverwaltung angefordert werden kann, die Gültigkeit der Signatur. Diese Schnittstelle bietet alle in Abschnitt A.6.5 beschriebenen Funktionen. Das dazugehörige WSDL-File und das XML-Schema zum Serialisieren der Requests befinden sich im Anhang (A.7.5).
  Wie schon bei der Schnittstelle zu normalen Clients wird hier auch das *wsDualHttpBinding* verwendet. Auch hier könnte bei Problemen mit nicht WCF-Clients auf einen Polling-Mechanismus umgestellt werden.

5.2.1 Speicherung der Daten in der Datenbank

In diesem Abschnitt wird darauf eingegangen, wie die Daten eines Kunden in einer Datenbank abgelegt werden. Bei der Demo-Applikation wurde eine *Microsoft SQL-Express* Datenbank verwendet. Abbildung 5.2 zeigt das verwendete Datenbankschema.
Die Tabelle *user* speichert die Daten eines Benutzers. Momentan wird lediglich der Benutzername (Spalte *userName*) des Benutzers gespeichert.

In der Tabelle *storage* werden die Datensätze, welche ein Kunde im Datenspeicher abgelegt hat, gespeichert. Dazu wird neben dem verschlüsselten Datensatz (Spalte *encryptedData*) der Verschlüsselungsalgorithmus (Spalte *encryptionAlgorithm*), mit welchem der Datensatz symmetrisch verschlüsselt wurde und die Art des Datensatzes (Spalte *type*) gespeichert. Die asymmetrisch verschlüsselten Session-Keys, mit welchen die Datensätze symmetrisch verschlüsselt wurden, werden in der Tabelle *keysForStorage* gespeichert (Spalte *encryptedSessionKey*). Auch hier wird wieder der Verschlüsselungsalgorithmus, mit welchem der Session-Key verschlüsselt wurde, gespeichert (Spalte *encryptionAlgorithm*). Außerdem wird die Schlüssel-ID des zugehörigen Schlüsselpaares hier gespeichert (Spalte *keyID*).

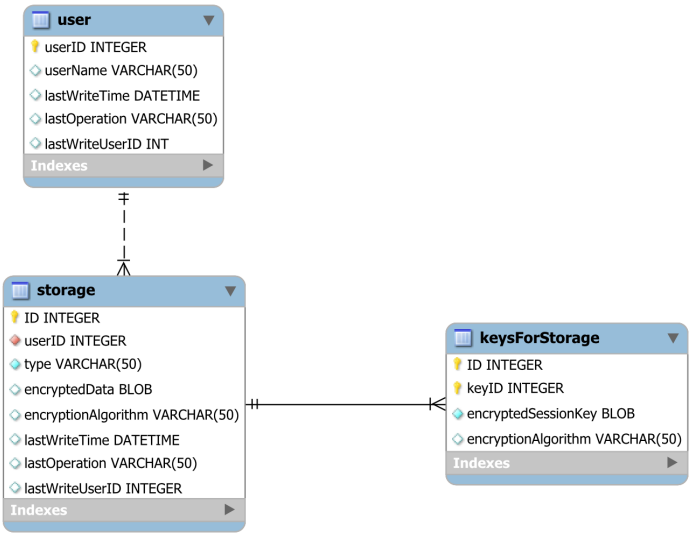


Abbildung 5.2: Datenbankschema von *Datenspeicher 1* und *Datenspeicher 2*

5.3 Komponente *Kundenclient*

Mithilfe des Kundenclients kann ein Kunde alle Use Cases, welche durch den Kunden betreffen werden, komfortabel ausführen. Der Kundenclient wurde als *Windows-Forms Anwendung* implementiert. Jeder Use Case wird über den entsprechenden Tab ausgewählt. Der gewünschte Datenspeicher wird rechts oben über die Drop-Down-Box ausgewählt. Im unteren Bereich des Fensters, kann der Kunde die Vorgänge, welche im Hintergrund passieren, genau verfolgen (Statusfenster). Eventuelle Fehlermeldungen werden ebenfalls hier ausgegeben. In Abbildung 5.3 ist ein Screenshot des Kundenclients zu sehen.

5.4 Komponente *Beraterclient*

Mithilfe des Beraterclients kann ein Berater alle Daten, welche er mit den für ihn freigegebenen Schlüsseln eines Kunden entschlüsseln kann, einsehen. Außerdem kann der Berater Daten bzw. Schlüssel bei einem Kunden anfordern. Wie der Kundenclient ist auch der Beraterclient als *Windows-Forms Anwendung* implementiert. Auch hier wird jede Funktionalität über den ensprechenden Tab gewählt. Im unteren Bereich des Fensters kann der Berater die Vorgänge, welche im Hintergrund passieren, genau verfolgen. Eventuelle Fehlermeldungen werden ebenfalls hier ausgegeben. Abbildung 5.4 zeigt einen Screenshot des Beraterclients.

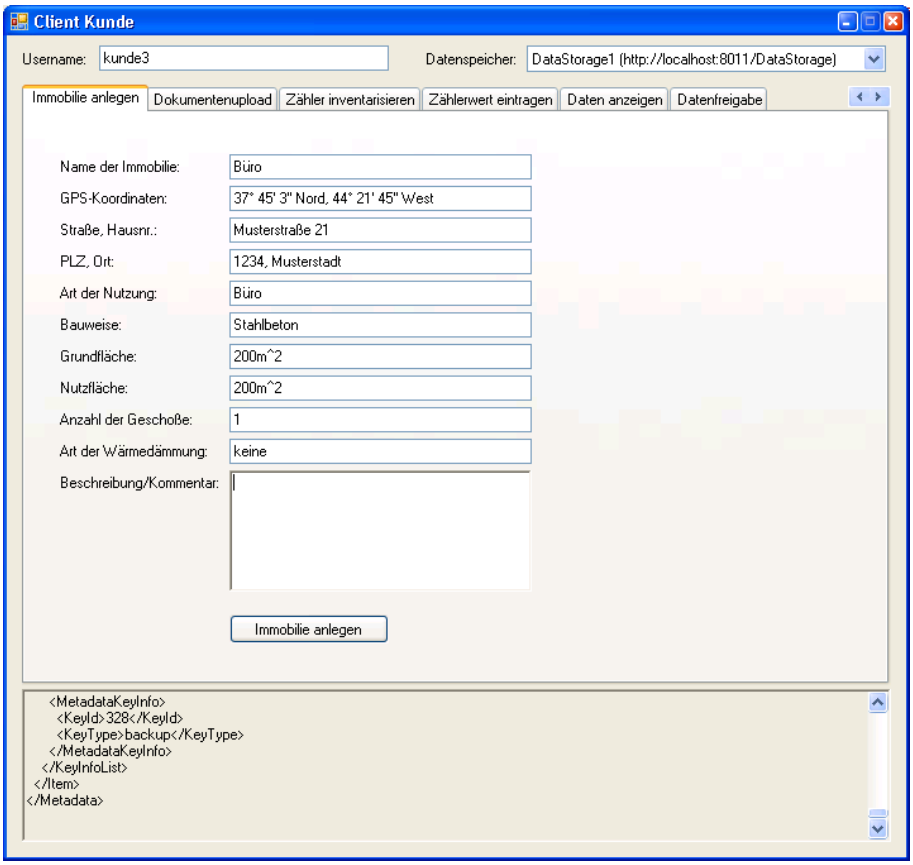


Abbildung 5.3: Screenshot des Kundenclients (Anlegen einer Immobilie)

5.5 Komponente *Automatischer Zähler*

Diese Komponente simuliert einen automatischen Zähler (automatische Datenquelle). Mit dieser *Windows-Forms Anwendung* kann ein Parameter-Datensatz geladen werden. Sobald die Parameterdaten geladen sind, kann mithilfe des *Start*-Buttons das Einbringen von Messdaten begonnen werden. Dabei wird so lange jede Minute ein Zufallswert als simulierter Messwert eingetragen, bis die Messung mithilfe des *Stop*-Buttons gestoppt wird. Im unteren Bereich des Fensters können die Vorgänge bzw. die eventuellen Fehlermeldungen verfolgt werden. In Abbildung 5.5 ist ein Screenshot der automatischen Datenquelle zu sehen.

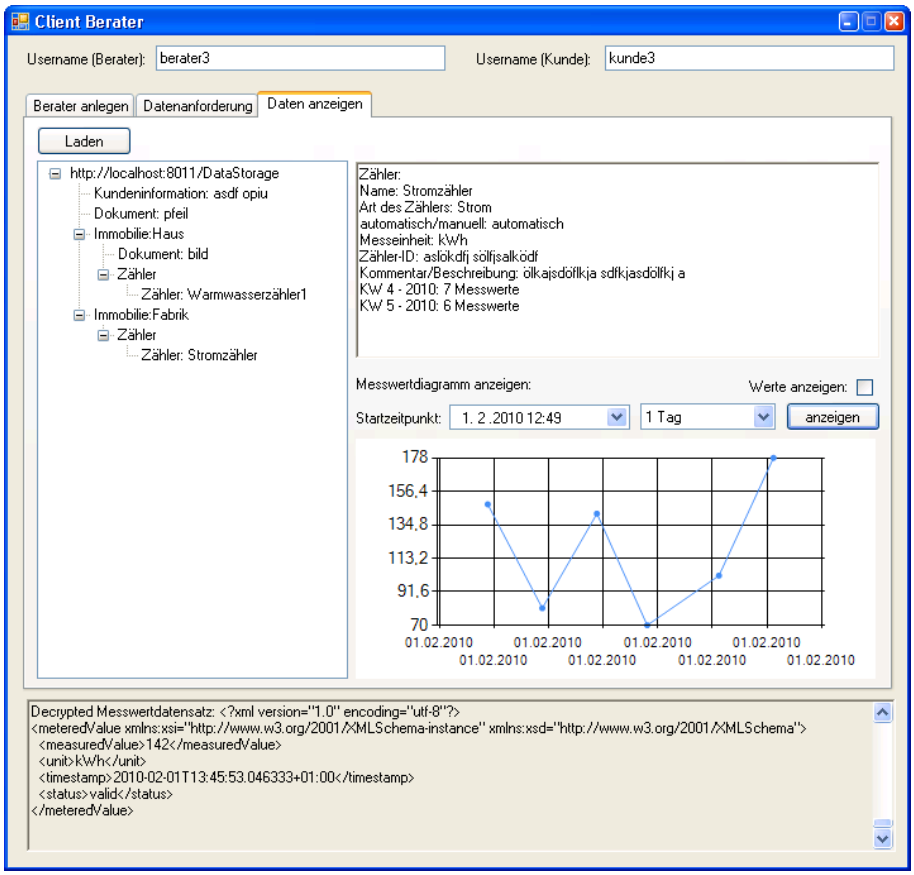


Abbildung 5.4: Screenshot des Beraterclients (Anzeige von Kundendaten)

5.6 Demonstration der Demo-Applikation anhand eines Sytemprozesses

In diesem Abschnitt wird anhand, des in Abschnitt A.5.4 beschriebenen Systemprozess „Upload eines Dokuments durch den Kunden“ mithilfe von Screenshots und Statusausgaben des Kundenclients die Funktionsweise der Demo-Applikation gezeigt:

Abbildung 5.6 zeigt nochmal den zu demonstrierenden Prozess. In rot sind die Schritte eingetragen, welche im weiteren näher erklärt werden.

Schritt 1: Der Kunde wählt den Menüpunkt zum Upload eines Dokuments: Abbildung 5.7 zeigt den Tab der durch den Kunden angeklickt wird, um die entsprechenden Funktion auszuführen.

Schritt 2: Als nächstes wählt der Kunde, ob es sich bei dem neuen Dokument um ein Kundendokument oder ein Immobiliendokument handelt. In diesem Fall wird von einem Immobiliendokument ausgegangen. Abbildung 5.7 zeigt den zu aktivierenden

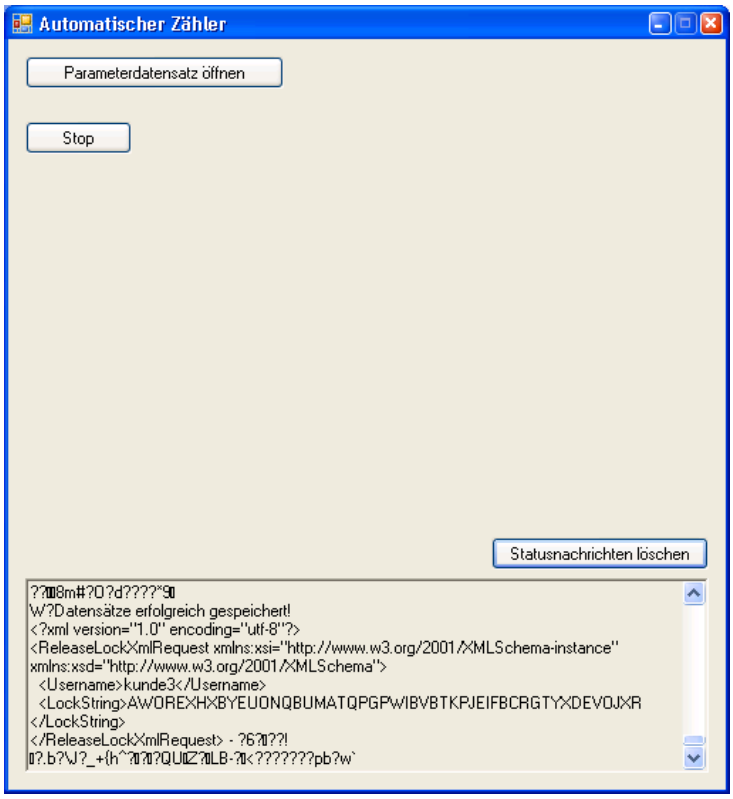


Abbildung 5.5: Screenshot des automatischen Zählers

Radio-Button.
Sobald dieser aktiviert wurde, lädt und entschlüsselt der Client den MetadataRoot-Datensatz, um dem Kunden eine Liste seiner, in diesem Datenspeicher gespeicherten, Immobilien anzuzeigen. Die entsprechende Ausgabe im Statusfeld des Kundenclients:

```
1  Response - Schlüsselverwaltung: 0 - Schlüsselliste wurde erfolgreich erstellt (Schlüssel-
       Anzahl: 1)
2  Response - Datenspeicher: 0 - Liste der Daten erfolgreich erstellt!
3  Response - Datenspeicher: 0 - Die Datensätze wurden erfolgreich gelesen!
4  Response - Schlüsselverwaltung: 0 - Holen des Private Key des Schlüsselpaares erfolgreich
       !
5  Decrypted Metadata-Datensatz: <?xml version="1.0" encoding="utf-8"?>
6  <Metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.
       org/2001/XMLSchema">
7          ...
8  </Metadata>
```

Als erstes fordert der Client bei der Schlüsselverwaltung eine Liste der Schlüssel des Kunden des Typs „backup“ an. Die Schlüsselverwaltung antwortet mit der entsprechenden Liste und dem Status 0 (erfolgreich). Anschließend fordert der Client eine Liste aller Datensätze des Typs „metadataRoot“ beim gewählten Datenspeicher an. Dieser antwortet mit einer Liste der Datensätze und dem Status 0, was *erfolgreich* bedeutet. Die Liste beinhaltet genau einen Eintrag. Der Client wählt die ID genau die-

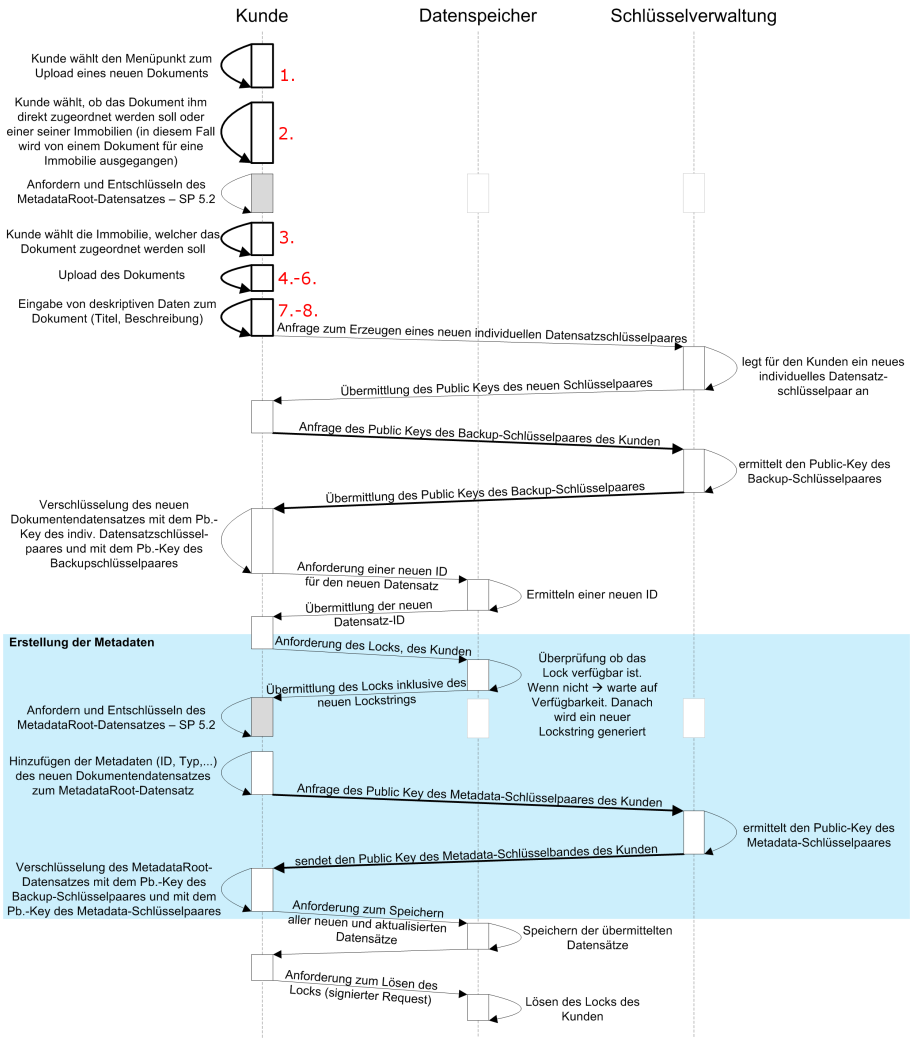


Abbildung 5.6: Use Case *Upload eines Dokuments durch den Kunden*

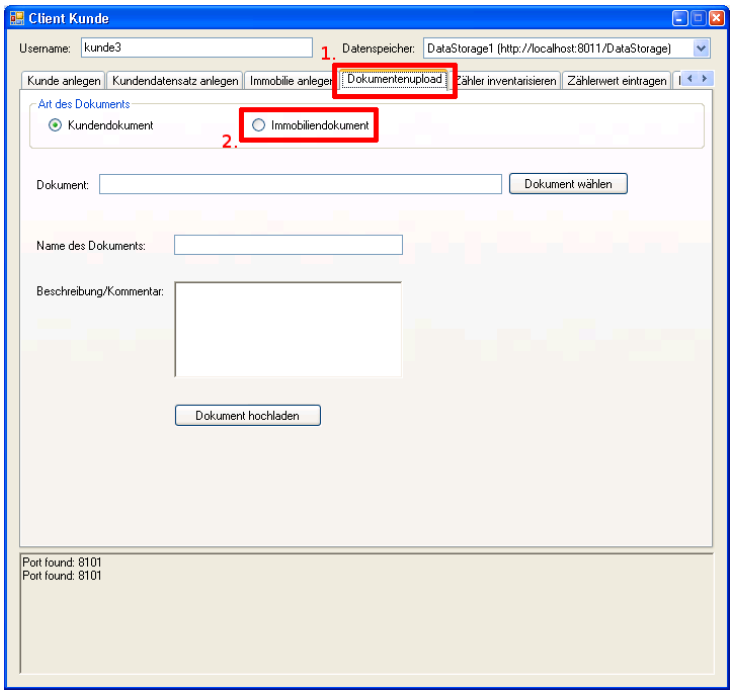


Abbildung 5.7: Kunde wählt den Menüpunkt zum Uploaden von Dokumenten; Kunde wählt Kundendokument oder Immobiliendokument (hier Immobiliendokument)

ses Datensatzes und fordert somit den MetadataRoot-Datensatz beim Datenspeicher an. Dieser antwortet mit dem geforderten Datensatz und dem Status 0 (erfolgreich). Schlussendlich fordert der Client den Private-Key des Backup-Schlüsselpaares bei der Schlüsselverwaltung an. Diese antwortet mit dem angeforderten Schlüssel (Status 0 - erfolgreich). Danach kann der Client den MetadataRoot-Datensatz entschlüsseln und somit dem Kunden eine Liste der gespeicherten Immobilien anzeigen.

Schritt 3: Der Kunde wählt aus der Liste der Immobilien die gewünschte Immobilie aus. Abbildung 5.8 zeigt diesen Schritt.

Schritt 4: Der Kunde wählt den Menüpunkt zur Auswahl des Dokuments aus (siehe Abbildung 5.9). Danach erscheint ein Dialogfenster zum Öffnen von Dateien.
Schritt 5: In diesem Dialogfenster wählt der Kunde das gewünschte Dokument aus (siehe Abbildung 5.9).
Schritt 6: Danach klickt der Kunde auf den *Öffnen*-Button (siehe Abbildung 5.9).
Schritt 7: Als nächstes gibt der Kunde die deskriptiven Daten zum neuen Dokument ein. In Abbildung 5.10 sieht man diesen Schritt.
Schritt 8: Danach klickt der Kunde auf den *Dokument hochladen*-Button (siehe Abbildung 5.10), um das Speichern des Dokuments zu bestätigen.

Anschließend beginnt der Client des Kunden alle notwendigen Schritte zum Speichern des Dokuments zu setzen. Dabei gibt der Client folgende Ausgaben aus, welche

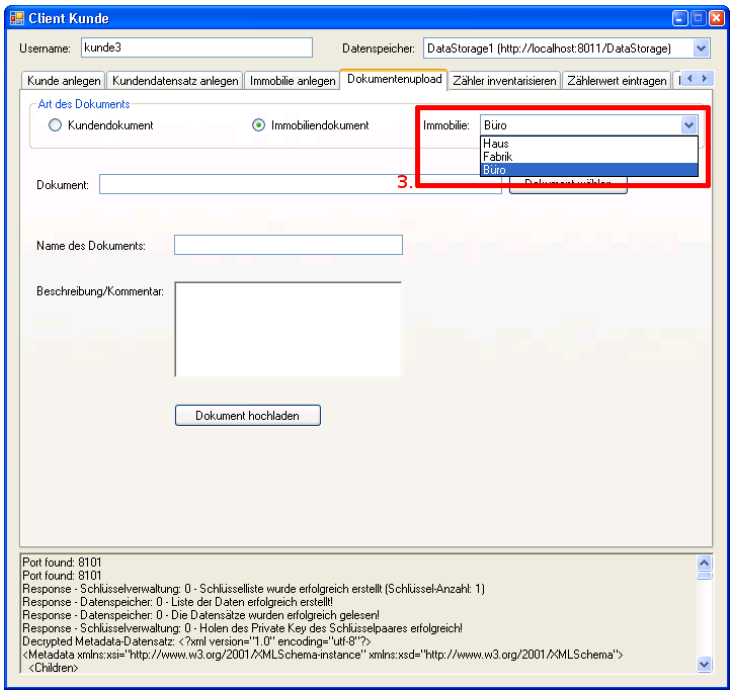


Abbildung 5.8: Kunde wählt die Immobilie, welcher das neue Dokument zugeordnet werden soll

anschließend näher erläutert werden:

```
1  ...
2  Zu sendender Datensatzstring: <?xml version="1.0" encoding="utf-8"?>
3  <document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.
       org/2001/XMLSchema">
4    <name>Dokument Büro</name>
5    <filename>Dokument Büro.doc</filename>
6    <comment>Beschreibung zum "Dokument_Büro"</comment>
7    <documentData>k48djAsdkl2dj30dH2km3hegI0dk3öa03jnk3</documentData>
8  </document>
9  Response - Schlüsselverwaltung: 0 - Neuer individueller Schlüssel wurde angelegt! (
       Schlüssel-ID: 395, Public Key: <RSAKeyValue><Modulus>1GwTisPB0Sze8bx0l9+1
       YXkBDZAjlee8EwXuGlkz/1bSIqvJGrAYb/21QbLTBVAavlChpZL0PuKi0YD1K/feSouUhFXmi/
       MI7YnDTkzQIniQykVvjDNwoXkmw2uar08kS42wlnla+N0gp9sHMPHa++PAeJZN7s/Zv78Hd077ssU=</
       Modulus><Exponent>AQAB</Exponent></RSAKeyValue>)
10 Response - Schlüsselverwaltung: 0 - Schlüsselliste wurde erfolgreich erstellt (Schlüssel-
       Anzahl: 1)
11 Response - Schlüsselverwaltung: 0 - Holen des Public Key des Schlüsselpaares erfolgreich!
        (Public Key: <RSAKeyValue><Modulus>qa439e0KSVt5RA8awNeg8hZL30TIj0+28
       tHs2ZvVbwKQWUqdPkX1h7cMDj/aYKm9LXkOXJ7xsY14vlV3Gv33oz5lT3FYiUDQ8C+19
       dI3evMTf2qGbnBDsIUmguRvwSOVhJFqu0/nD1vUVu2xiqYjTiXyysE9hug3rE7ET+nk8kk=</Modulus><
       Exponent>AQAB</Exponent></RSAKeyValue>)
12 ...
```

Als erstes wird der Datensatz mit dem neuen Dokument ins XML-Format serialisiert. Danach wird bei der Schlüsselverwaltung ein neuer individueller Datensatzschlüssel für das neue Dokument angefordert. Die Schlüsselverwaltung antwortet mit dem Public-Key des neuen Schlüsselpaares (Status 0 - erfolgreich).

Danach wird eine Liste aller Schlüssel des Kunden mit dem Typ „backup“ bei der

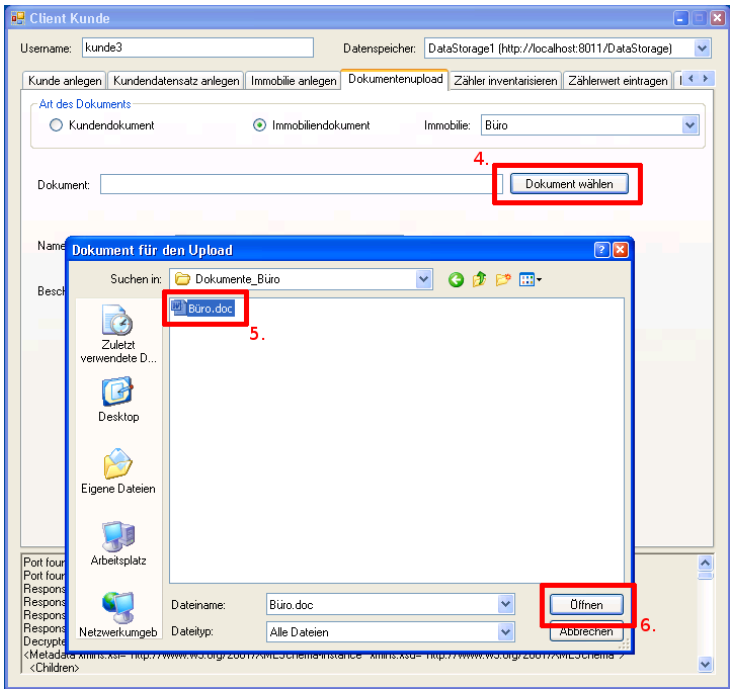


Abbildung 5.9: Kunde wählt das hochzuladende Dokument aus

Schlüsselverwaltung angefordert. Diese antwortet mit einer Liste von Schlüsselinformationen mit einem Eintrag (Status 0 - erfolgreich). Anschließend wird der Public-Key des Backupschlüsselpaares bei der Schlüsselverwaltung angefordert, welche diesen mit dem Status 0 zurückliefert.

Anschließend wird der serialisierte Dokumentendatensatz symmetrisch mit einem neu generierten Session-Key verschlüsselt und dieser Session-Key anschließend sowohl mit dem Public-Key des neu erzeugten Schlüsselpaares, als auch mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Zu diesem Schritt gibt es keine Ausgabe im Statusfenster.

```
1  ...
2  LockString sofort erhalten: VIATFBPXRLWPDSSCHVMVPHJNMCTOCMHAVGMHEMIYMGEROLFXID!
3  Response - Schlüsselverwaltung: 0 - Schlüsselliste wurde erfolgreich erstellt (Schlüssel-
       Anzahl: 1)
4  Response - Datenspeicher: 0 - Liste der Daten erfolgreich erstellt!
5  Response - Datenspeicher: 0 - Die Datensätze wurden erfolgreich gelesen!
6  Response - Schlüsselverwaltung: 0 - Holen des Private Key des Schlüsselpaares erfolgreich
       !
7  Decrypted Metadata-Datensatz: <?xml version="1.0" encoding="utf-8"?>
8  <Metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.
       org/2001/XMLSchema">
9    <Children>
10     <Metadata>
11       <Children />
12       <ChildrenMetadataSets />
13       <Item>
14         <Type>facility</Type>
15         <Name>Büro</Name>
16         <DataId>319</DataId>
```

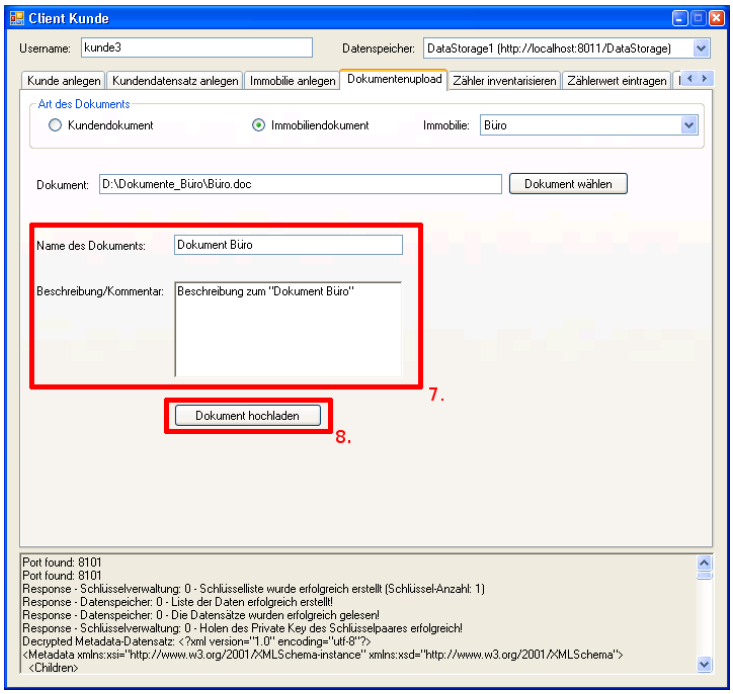


Abbildung 5.10: Eingabe von deskriptiven Daten zum Dokument

```
17          <WeekInYear>0</WeekInYear>
18          <Year>0</Year>
19          <SignatureKeyId>0</SignatureKeyId>
20          <KeyInfoList>
21            <MetadataKeyInfo>
22              <KeyId>390</KeyId>
23              <KeyType>individual</KeyType>
24            </MetadataKeyInfo>
25            <MetadataKeyInfo>
26              <KeyId>328</KeyId>
27              <KeyType>backup</KeyType>
28            </MetadataKeyInfo>
29          </KeyInfoList>
30        </Item>
31      </Metadata>
32    </Children>
33    <ChildrenMetadataSets />
34  </Metadata>
35  Zu sendender Datensatzstring: <?xml version="1.0" encoding="utf-8"?>
36  <Metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.
        org/2001/XMLSchema">
37    <Children>
38      <Metadata>
39        <Children>
40          <Metadata>
41            <Item>
42              <Type>document</Type>
43              <Name>Dokument Büro</Name>
44              <DataId>324</DataId>
45              <WeekInYear>0</WeekInYear>
46              <Year>0</Year>
47              <SignatureKeyId>0</SignatureKeyId>
48              <KeyInfoList>
49                <MetadataKeyInfo>
```

```
50                  <KeyId>395</KeyId>
51                  <KeyType>individual</KeyType>
52                </MetadataKeyInfo>
53                <MetadataKeyInfo>
54                  <KeyId>328</KeyId>
55                  <KeyType>backup</KeyType>
56                </MetadataKeyInfo>
57              </KeyInfoList>
58            </Item>
59          </Metadata>
60        </Children>
61        <ChildrenMetadataSets />
62        <Item>
63          <Type>facility</Type>
64          <Name>Büro</Name>
65          <DataId>319</DataId>
66          <WeekInYear>0</WeekInYear>
67          <Year>0</Year>
68          <SignatureKeyId>0</SignatureKeyId>
69          <KeyInfoList>
70            <MetadataKeyInfo>
71              <KeyId>390</KeyId>
72              <KeyType>individual</KeyType>
73            </MetadataKeyInfo>
74            <MetadataKeyInfo>
75              <KeyId>328</KeyId>
76              <KeyType>backup</KeyType>
77            </MetadataKeyInfo>
78          </KeyInfoList>
79        </Item>
80      </Metadata>
81    </Children>
82    <ChildrenMetadataSets />
83  </Metadata>
84  Response - Schlüsselverwaltung: 0 - Schlüsselliste wurde erfolgreich erstellt
85  Response - Schlüsselverwaltung: 0 - Holen des Public Key des Schlüsselpaares erfolgreich!
86  Die Datensätze wurden erfolgreich gespeichert!
```

Als nächstes wird beim Datenspeicher eine neue ID für den Dokumentendatensatz angefordert. Der Datenspeicher reserviert eine neue ID für den Kunden und liefert diese zurück. Für diesen Schritt gibt es keine Ausgabe im Statusfenster.

Danach fordert der Kunde das Lock beim Datenspeicher an. Dieses ist sofort verfügbar und der Datenspeicher liefert den eindeutigen Lockstring des Locks zurück. Anschließend wird nochmals der MetadataRoot-Datensatz gelesen. Dies geschieht in der selben Reihenfolge wie zuvor.

Anschließend werden die Metadaten des neuen Dokumentendatensatzes in den MetadataRoot-Datensatz eingetragen. In der Ausgabe sieht man die relevanten Auschnitte des Datensatzes vor und nach dem Hinzufügen dieser Metadaten.

Als nächstes wird bei der Schlüsselverwaltung eine Liste aller Schlüssel des Kunden mit dem Typ „metadata“ angefordert. Dies wird mit einer Liste mit genau einem Schlüssel beantwortet. Danach fordert der Client des Kunden den Public-Key dieses Schlüsselpaares an. Die Schlüsselverwaltung liefert den angeforderten Public-Key mit dem Status 0.

Nun kann der Client den MetadataRoot-Datensatz mit einem neu generierten Session-Key verschlüsseln. Anschließend wird dieser Session-Key mit dem Public-Key des Metadata-Schlüsselpaares und dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Zu diesem Schritt wird keine Ausgabe angezeigt.

Nachdem der MetadataRoot-Datensatz neu verschlüsselt wurde, werden sowohl der Dokumentendatensatz, als auch der MetadataRoot im Datenspeicher gespeichert.

Abschließend wird das Lock des Kunden gelöst (keine Ausgabe).

5.7 Zusammenfassung

In diesem Kapitel wurde die Demo-Applikation vorgestellt, welche unter anderem als Beweis für die Vollständigkeit und Korrektheit des Designkonzepts dient. Diese Demo-Applikation wurde in *Visual Studio 2008* entwickelt und bedient sich der Programmierschnittstelle *WCF*, welche eine einfache Entwicklung von Web Services ermöglicht. Die Demo Applikation besteht aus folgenden Komponenten: Schlüsselverwaltung, Datenspeicher 1+2, Kundenclient, Beraterclient und einer automatischen Datenquelle. Es wurden alle Server- und Client-Komponenten vorgestellt, wobei bei den Server-Komponenten auch auf deren Schnittstellen und die Speicherung der Daten in einer SQL-Datenbank eingegangen wurde. Abschließend wurde die Funktionsweise der Demo-Applikation anhand eines Systemprozesses demonstriert.

KAPITEL 6

Diskussion

In diesem Kapitel wird nun das in Kapitel 4 vorgestellte Designkonzept diskutiert. Dazu wird als erstes dargelegt, wie die Daten im Datenspeicher konsistent gehalten werden bzw. ob Fehler während eines schreibenden Systemprozesses eine Auswirkung auf diese Konsistenz haben. Danach wird noch einmal die Verschlüsselung der Metadaten diskutiert. Dabei wird untersucht, wie sehr sich die Systemprozesse durch die Verschlüsselung der Metadaten verkomplizieren und wie einfach der selbe Prozess aussehen könnte, würden diese unverschlüsselt gespeichert. Im darauffolgenden Abschnitt wird die Einschränkung der Parallelität durch den Lock-Mechanismus aufgezeigt und ein Lösungsvorschlag die Problematik vorgestellt. Am Ende dieses Kapitels wird der Speicherbedarf durch die Speicherung der Daten eines Kunden analysiert. Dabei wird sowohl die nicht historisierte Speicherung als auch die historisierte Speicherung diskutiert.

6.1 Konsistenzhaltung der Daten im Datenspeicher

In diesem Abschnitt wird diskutiert, wie die Daten, welche im Datenspeicher abgelegt werden, konsistent gehalten werden. Es soll erklärt werden, wie sichergestellt werden kann, dass zu jedem gespeicherten Datensatz auch ein Metadatensatz exisistiert, welcher diesen Datensatz beschreibt bzw. auf ihn verweist. Außerdem muss sichergestellt werden, dass ein Datensatz, zu welchem Metadaten verfügbar sind, auch wirklich existiert und die entsprechenden Verweise gültig sind.

Prinzipiell ist festzustellen, dass durch die Verschlüsselung der Metadaten die Konsistenz allein durch den Datenspeicher nicht sichergestellt werden kann, da dieser die Metadaten nicht lesen und verarbeiten kann. Dies bedeutet, dass ein großer Teil der Konsistenz durch den Kundenclient und durch die Clients der automatischen Datenquellen eines Kunden sichergestellt werden muss (das sind jene Clients welche Daten manipulieren, d.h. schreibend auf den Datenspeicher zugreifen). Dies bedeutet, dass die korrekte Implementierung der Clients mindestens genauso wichtig ist, wie die

korrekte Implementierung der Services. Der Client muss also garantieren, dass, wann immer ein schreibender Zugriff auf den Datenspeicher stattfindet, die entsprechenden Metadaten ebenso aktualisiert und gespeichert werden.

Jedoch kann auch der Client allein die Konsistenz nicht sicherstellen. So muss erstens sichergestellt werden, dass zu jedem Zeitpunkt für jeden Kunden nur ein Client die Metadaten, welche zu aktualisieren sind, liest, diese aktualisiert und schließlich aktualisiert wieder auf einen Datenspeicher zurückschreibt. Aus diesem Grund wurde der in Abschnitt 4.6 beschriebene Lock Mechanismus eingeführt, welcher sicherstellt, dass ein Client für die Zeit der Bearbeitung eines Datensatzes und der Aktualisierung der zugehörigen Metadaten exklusiven Zugriff auf diesen Datenspeicher hat. Alle Vorgänge zwischen dem Anfordern des Locks und dem Speichern der neuen und aktualisierten Datensätze sind in den Systemprozessen (Abschnitt 4.10 bzw. Abschnitt A.5) durch einen blauen Hintergrund hervorgehoben. Innerhalb dieses Bereichs findet auch immer die Aktualisierung der Metadaten statt.

Zusätzlich muss sichergestellt werden, dass beim Aktualisieren oder neu Anlegen eines Datensatzes alle Schreibzugriffe (also auch die Aktualisierung der Metadaten) in einem Funktionsaufruf beim Datenspeicher abgehandelt werden. Diese Funktion ist die Funktion *AddUpdateData* für den Kundenclient und die Funktion *AddUpdateDataForAutomaticMeter* für automatische Datenquellen. Beide Funktionen führen alle übergebenen Schreibzugriffe als eine Transaktion aus. Dies führt dazu, dass, wenn ein Fehler während der Bearbeitung auftritt, der Vorgang so abgebrochen werden kann, als wäre er nie passiert. Die Datenbank wird also wieder in den Zustand vor dem Aufruf zurückversetzt (Rollback). Wenn alle Operationen erfolgreich durchgeführt werden, werden alle Änderungen dieses Aufrufs gemeinsam festgeschrieben (Commit). Somit können alle Schreibzugriffe, welche für die Einbringung oder Aktualisierung eines Datensatzes notwendig sind, atomar durchgeführt werden. Es werden also entweder alle oder gar keine Daten im Datenspeicher geschrieben. Für die Richtigkeit der Daten ist allerdings weiterhin der Client verantwortlich.

Da alle Schreibzugriffe eines Systemprozesses in einem Aufruf stattfinden müssen, aber die ID eines neuen Datensatzes im zugehörigen Metadatensatz, schon eingetragen werden muss, muss die zukünftige ID des neuen Datensatzes schon vor dem Speichern feststehen. Aus diesem Grund wird jedes Mal, wenn ein neuer Datensatz angelegt werden soll, zuerst eine neue ID für diesen Datensatz beim Datenspeicher angefordert. Der Datenspeicher reserviert diese ID für diesen Kunden und markiert diese bis zur Speicherung des Datensatzes als unbenutzt.

Weiters sei erwähnt, dass durch die richtige Implementierung der Systemprozesse Deadlocks durch den Lock-Mechanismus ausgeschlossen sind, da in allen Situationen, in welchen ein Lock angefordert wird, dies immer nur bei einem einzigen Datenspeicher passiert.

Der Leser hat sich möglicherweise schon gefragt, warum bei einigen Systemprozessen die selben Metadaten mehrmals gelesen und entschlüsselt werden. Dabei werden diese Datensätze stets einmal anfangs zur Anzeige von Listen für den Kunden gelesen. Anschließend gibt der Kunde die Daten ein und schickt diese abschließend ab. Da sich die Metadatensätze inzwischen geändert haben könnten und der Kundenclient erst nach dem Abschicken der Daten das Lock anfordert, muss der selbe Metadatensatz noch einmal gelesen werden, um immer mit den aktuellen Metadatensätzen zu

hantieren und somit die Konstistenz sicherzustellen.

Neben der Konsistenz der Daten sollte nicht undiskutiert bleiben was passiert, wenn während eines Systemprozesses ein Fehler auftritt. Dabei sind nur jene Fehler relevant, welche nach dem Abschicken der Daten durch den Kunden bzw. nach dem Beginn eines Use Cases zur Einbringung von Messdaten durch eine automatische Datenquelle auftreten. Alle Fehler, die hier auftreten können, können lediglich eine von zwei Arten von Auswirkungen haben. Es können einerseits individuelle Datensatzschlüssel angelegt worden sein, welche später niemals benutzt werden, da beispielsweise bei der Speicherung ein Fehler auftritt. Andererseits könnten Datensatz-IDs reserviert worden sein, welche ebenfalls niemals benutzt werden. Beide Arten führen zwar zu einem Resourcenverbrauch im Datenspeicher bzw. in der Schlüsselverwaltung, aber sie führen keinesfalls zu Inkonsistenzen. Einzig nachteilig zu erwähnen ist, dass ein Cleanup solcher unnötig reservierter Resourcen einen schwierigen Prozess darstellt, da er vor allem bei den unnötig angelegten Schlüsseln nicht allein von der Schlüsselverwaltung durchgeführt werden kann. Der Grund hierfür ist, dass die Schlüsselverwaltung keine Kenntnis darüber hat, ob tatsächlich ein Datensatz in einem Datenspeicher existiert, welcher mit diesem Schlüssel verschlüsselt wurde.

6.2 Verschlüsselte Metadaten vs. Unverschlüsselte Metadaten

In diesem Kapitel wird gezeigt, wie sehr die Verschlüsselung der Metadaten die Komplexität des Systems erhöht bzw. das System durch diese Verschlüsselung eingeschränkt wird.

Zuerst wird anhand des in Abschnitt A.5.3 vorgestellten Use Case „Automatische Messdatenerfassung (online)“ beispielhaft die Vorgangsweise im Fall unverschlüsselter Metadaten mit jener im Fall verschlüsselter Metadaten verglichen.

Im Fall der unverschlüsselten Metadaten werden zwar die Datensätze selbst verschlüsselt gespeichert. Alle anderen Daten, wie zum Beispiel die Art und der Name des Datensatzes, werden im Datenspeicher aber unverschlüsselt zusammen mit dem Datensatz gespeichert.

Abbildung 6.1 zeigt den Use Case im Fall unverschlüsselter Metadaten. Abbildung 6.2 zeigt nochmals die schon in Abschnitt A.5.3 vorgestellte verschlüsselte Version.

Beschreibung: Use Case: Automatische Messdatenerfassung (online) - unverschlüsselt

Nachdem die automatische Datenquelle den Messwert und den zugehörigen Zeitpunkt der Messung ermittelt hat, wird überprüft, ob der vorhandene Wochenschlüssel noch aktuell ist. Hier wird davon ausgegangen, dass dies nicht der Fall ist und somit ein neuer Wochenschlüssel bei der Schlüsselverwaltung angefordert werden muss. Also wird der Public-Key des aktuellen Wochenschlüssels des in der Datenquelle parametrierten Schlüsselbandes bei der Schlüsselverwaltung angefordert. Nachdem die Schlüsselverwaltung den angeforderten Schlüssel zurückgeliefert hat, erstellt die au-

Systemprozess: 1.1 Automatische Messdatenerfassung (online)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar)
Kurzbeschreibung: Die Datenquelle fordert bei der Schlüsselverwaltung einen neuen Wochenschlüssel an (weil bspw. der alte abgelaufen ist). Danach werden die Daten mit dem aktuellen Wochenschlüssel und mit dem Public Key des Backup-Schlüsselpaares verschlüsselt und mit dem Privaten Schlüssel der Quelle signiert. Anschließend werden die Daten an den Datenspeicher geschickt. Dort wird die Signatur geprüft und die Daten gespeichert.

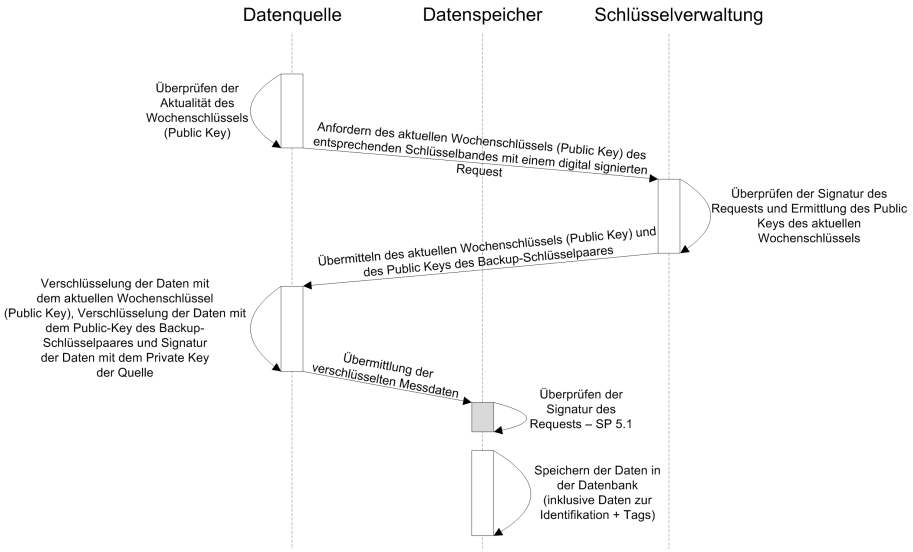


Abbildung 6.1: Use Case: Automatische Messdatenerfassung (online) - **unverschlüsselt**

tomatische Datenquelle den Messwertdatensatz und verschlüsselt diesen mit einem neuen Session-Key. Danach wird dieser Session-Key mithilfe des Public-Keys des Backup-Schlüsselpaares des Kunden und mit dem soeben gelieferten Public-Key des Wochenschlüsselpaares verschlüssselt. Danach wird der neue Datensatz mit einem digital signierten Request an den Datenspeicher zum Speichern übermittelt. Der Datenspeicher überprüft die Signatur des Requests und speichert anschließend den Datensatz in der Datenbank.

Vergleicht man die Abläufe dieses Use Cases im verschlüsselten und im unverschlüsselten Fall, so erkennt man ohne Zweifel, dass die Abläufe im unverschlüsselten Fall einfacher und daher auch schneller erledigt werden können als im verschlüsselten Fall. Außerdem sei gesagt, dass die in Abschnitt 6.1 beschriebenen Maßnahmen zur Konsistenzhaltung im unverschlüsselten Fall nicht mehr notwendig sind, da die Metadaten direkt an den Datensatz geknüpft sind und daher auch gemeinsam gespeichert werden. Dies bedeutet, dass sowohl der Lock-Mechanismus als auch die Anforderung neuer Datensatz-IDs im unverschlüsselten Fall nicht gebraucht wird. Dies hätte auch den Vorteil, dass kein Callback mehr benötigt werden würde. Somit wäre die Interopera-

Systemprozess: 1.1 Automatische Messdatenerfassung (online)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar); die Datenquelle hat einen neuen Messwert und den zugehörigen Messzeitpunkt; die Datenquelle ist richtig parametriert
Kurzbeschreibung: Die Datenquelle fordert bei der Schlüsselverwaltung einen neuen Wochenschlüssel an. Danach werden die Daten mit dem aktuellen Wochenschlüssel und mit dem Public Key des Backup-Schlüsselpaares verschlüsselt und beim Datenspeicher eine neue Datensatz-ID angefordert. Nachdem die Datenquelle das Kunden-Lock angefordert hat, liest diese den Metadatensatz des Zählers und legt neue Metadatensätze an bzw. aktualisiert diese (trägt den neuen Messwert ein). Danach werden die Daten im Datenspeicher gesichert und das Lock gelöst.

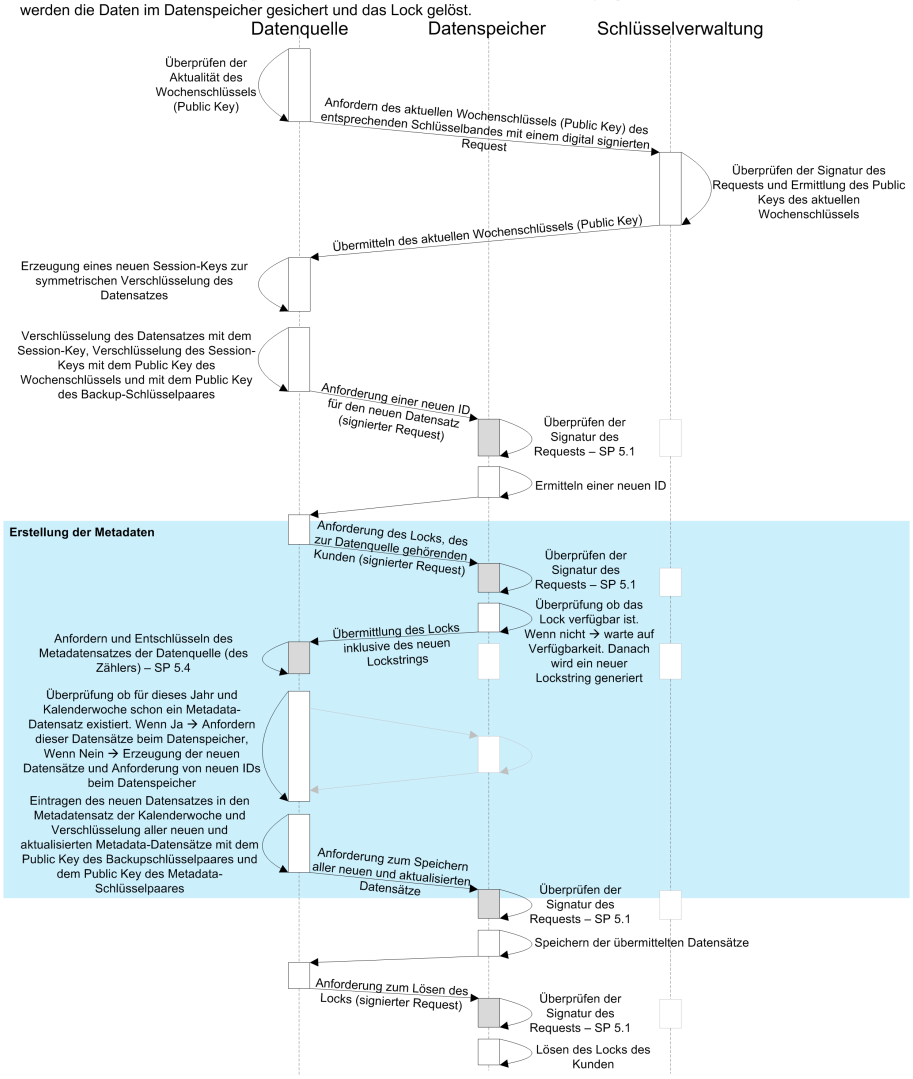


Abbildung 6.2: Use Case: Automatische Messdatenerfassung (online) - **verschlüsselt**

bilität des Systems wesentlich erhöht. Zwar könnte wie in Abschnitt 4.6 beschrieben, das Callback durch einen Polling-Mechanismus eliminiert werden. Allerdings ist dies wenig elegant und erhöht außerdem den Traffic auf der Leitung und verzögert somit die Vorgänge unnötig. Zusätzlich würden im unverschlüsselten Fall die Wartezeiten auf das Lock entfallen, was die Parallelität erhöht. D.h. es wäre unerheblich, wie viele automatische Datenquellen eines Kunden zeitgleich Messdaten einliefern wollen. Im Falle des einfachen Lock-Mechanismusses, wie er in der in Abschnitt 5 beschriebenen Demo-Applikation angewendet wurde, kann das einbringen von Messdaten immer nur seriell abgearbeitet werden. D.h. es kann zu jedem Zeitpunkt für jeden Kunden pro Datenspeicher immer nur ein Client Daten einbringen.

Ein weiterer Punkt ist, dass im Falle unverschlüsselter Metadaten ein völliger Offline-Betrieb möglich wäre. Eine automatische Datenquelle, welche beispielsweise über eine Speicherkarte mit den aktuellen Public-Keys versorgt werden würde, könnte die verschlüsselten Messdatensätze ebenfalls auf diese Speicherkarte schreiben. So könnte der Kunde beispielsweise einmal im Monat die Speicherkarte tauschen, um die automatische Datenquelle mit den Schlüsseln für die nächsten Wochen zu versorgen und um die Daten, welche seit dem letzten Kartentausch aufgezeichnet wurden, abzuholen. Danach kann der Kunde die Daten über einen Client, welcher eine Verbindung zum Datenspeicher hat, im Datenspeicher ablegen. Der Vorteil wäre, dass die Messwertdatensätze im Falle unverschlüsselter Metadaten schon verschlüsselt auf der Speicherkarte vorliegen könnten. Nur die Metadaten sind also lesbar.

Bei verschlüsselten Metadaten ist so eine Vorgangsweise, bei welcher die Daten, welche später im Datenspeicher gespeichert werden, schon fertig auf der Speicherkarte vorliegen, unmöglich. Der Grund hierfür ist, dass der Metadatensatz zur Eintragung der Metadaten vom Datenspeicher geladen werden muss, was aber für die automatische Datenquelle, welche keine Verbindung zum Internet hat, unmöglich ist. Die einzige Möglichkeit wäre, wie im Fall der unverschlüsselten Metadaten, auf der Speicherkarte zwar die Messwertdatensätze zu verschlüsseln, aber die Metadaten unverschlüsselt zu lassen. Das Eintragen und Verschlüsseln der Metadaten wird beim Upload der Daten in den Datenspeicher nachgeholt. Das heißt, die Daten können nicht wie im unverschlüsselten Fall direkt an den Datenspeicher übergegeben werden, sondern erfordern eine Nachbearbeitung.

An dieser Stelle soll neben Einschränkungen, welche durch die Verschlüsselung der Metadaten entstehen, noch einmal gesagt werden, dass die verschlüsselte Version den Vorteil bietet, dass der Anbieter des Datenspeichers keine Schlüsse auf die Art der Daten, welche durch den Kunden im Datenspeicher abgelegt wurden, gezogen werden können. So kann der Datenspeicheranbieter (Hosting Provider) aus den Daten allein nicht erkennen wieviele Immobilien oder Zähler ein Kunde angelegt hat.

6.3 Verbesserung des Lock-Mechanismus

Da durch den in Abschnitt 4.6 beschriebenen Lock-Mechanismus die Parallelität sehr eingeschränkt ist, soll hier kurz eine Idee skizziert werden, welche den Lock-Mechanismus so verbessert, dass die Anzahl der automatischen Datenquellen eines Kunden,

welche in den selben Datenspeicher Daten einliefern, keinen Einfluss auf die Wartezeit auf das Lock mehr aufweisen.

Beim ursprünglichen Lock-Mechanismus (siehe Abschnitt 4.6) kann zu jedem Zeitpunkt für jeden Kunden immer nur ein Client Daten einbringen, da nur ein Client das aktuelle Lock halten kann. Dies führt, vor allem, wenn ein Kunde sehr viele automatische Zähler hat, welche in den gleichen Datenspeicher ihre Daten einliefern, zu langen Verzögerungen, bis ein Zähler das Lock erhält. Da allerdings jeder automatische Zähler unterschiedliche Metadatensätze bearbeitet, kann die automatische Messdatenerfassung getrost parallel erfolgen. Dies bedeutet, dass im Falle der automatischen Messdatenerfassung mehreren automatischen Datenquellen gleichzeitig ein Lock gegeben werden könnte.

Diese Idee führt dazu, dass zwei verschiedene Arten von Locks angewendet werden. Zum einem ein sogenanntes *Kundenlock*, welches dem ursprünglichen Lock sehr ähnelt und nur an einen Client vergeben werden darf. Das *Kundenlock* wird immer dann angefordert, wenn der Kunde über den Kundenclient Daten manipuliert oder einbringt. Wann immer dieses *Kundenlock* angefordert wird, muss solange gewartet werden, bis alle Locks des Kunden wieder freigegeben wurden. Erst danach erhält der Kundenclient das angeforderte *Kundenlock*. Danach muss jede Anforderung eines Locks des Kunden, unabhängig von der Art, solange verzögert werden, bis dieses *Kundenlock* wieder freigeben wurde.

Die zweite Art von Lock wird *Datensatzlock* genannt. Es sperrt lediglich den Metadatensatz des Zähler-Datensatzes. Da jeder Zähler einen eigenen Medadatensatz hat, kann im Normalfall das Lock sofort erteilt werden, sofern nicht das Kundenlock angefordert oder aktiv ist. Ein Konflikt - also wenn zwei automatische Datenquellen das gleiche *Datensatzlock* anfordern - kann nur auftreten, wenn automatische Datenquellen versehentlich gleich parametriert wurden. Dieser Fall sollte zwar nicht eintreten, führt allerdings neben der eventuellen Wartezeit auf das *Datensatzlock* und den sinnlosen Daten, welche durch den Fehler eingebracht werden, zu keinen weiteren Komplikationen.

Dies bedeutet, dass alle automatischen Zähler eines Kunden gleichzeitig Daten einbringen können, indem sie das *Datensatzlock* des Metadatensatzes des Zählers anfordern, die entsprechenden Metadaten aktualisieren und alle neuen und aktualisierten Datensätze im Datenspeicher sichern. Abschließend wird das *Datensatzlock* wieder gelöst. Wenn der Kunde über den Kundeclient Daten einbringen oder aktualisieren möchte, so fordert der Kundenclient das *Kundenlock* an, welches dem Kunden den exklusiven Zugriff auf alle ihm zugehörigen Daten in diesem Datenspeicher sichert. Danach werden wiederum die entsprechenden Metadaten angefordert und aktualisiert. Anschließend werden alle neuen und akualisierten Datensätze im Datenspeicher gesichert und das *Kundenlock* wieder gelöst. Das Kundenlock wird erst erteilt, wenn alle aktiven *Datensatzlocks* gelöst wurden. Während ein *Kundenlock* auf die Aktivierung wartet, werden keine keine neuen *Datensatzlocks* erteilt. Diese werden erst wieder erteilt, sobald das *Kundenlock* wieder gelöst wurde.

6.4 Speicherbedarf

In diesem Kapitel soll gezeigt werden, welchen Speicherbedarf die Speicherung der Daten, welche ein Kunde im Datenspeicher ablegt, aufweist. Dabei wird zuerst von einer nicht historisierten Speicherung ausgegangen und später die historisierte Speicherung untersucht. Die historisierte Datenspeicherung ist deswegen hier so interessant, da durch die Verschlüsselung der Metadaten sehr viele Update-Operationen auf den Metadatensätzen auftreten und diese bei der historisierten Datenspeicherung jedes mal einen neuen Datensatz in der Datenbank verursachen. Dieser Metadata-Datensatz ist bei der Einbringung neuer Daten stets größer als jener der Vorgängerversion.

Es wird auf jede Art von Datensätzen eingegangen und dabei werden typische Datensatzgrößen herangezogen. Dabei wird das Kommentarfeld stets leer gelassen, da dieses schwer einzuschätzen ist. Da sich der Speicherbedarf durch Eintragen eines Kommentars genau um die Länge des Kommentars verlängert (also mit dem Faktor 1), kann sehr einfach errechnet werden, welche Größen der Datensätze sich mit befüllten Kommentarfeld ergeben. Dokumentendatensätze werden ebenfalls etwas gesondert gehandhabt, da die Größe der Dokumente sehr stark variieren kann.

Ein sehr großer Teil des Speicherverbrauchs wird durch die Messdatensätze und deren Metadaten eingenommen. Diese lassen sich aufgrund der wenig variablen Felder sehr gut einschätzen. Dies bedeutet, dass die Schätzung bei diesen Daten sehr zuverlässig durchgeführt werden kann. Da die Metadatensätze zu den Messdaten sich stets über eine Woche erstrecken, kann nach der Beobachtung einer Woche der Speicherverbrauch auf einen längeren Zeitraum, wie zum Beispiel ein Jahr, hochgerechnet werden.

Am Ende des Kapitels wird noch kurz auf den Speicherbedarf in der Datenbank der Schlüsselverwaltung eingegangen. Diese Werte sind ebenfalls sehr stabil, da die Schlüssel immer die gleiche Länge aufweisen. In der Schlüsselverwaltung würde die historisierte Speicherung der Daten keinen großen Unterschied im Speicherbedarf ausmachen, da Schlüssel nie geändert werden, und somit die Datensätze immer nur in der ursprünglichen Version gespeichert werden.

Jeder in der Datenbank des Datenspeichers gespeicherte Datensatz verursacht neben dem serialisierten und verschlüsselten eigentlichen Datensatz noch zusätzliche Daten, welche folgenden Speicherbedarf hervorrufen:

- ID - 4 Bytes
- userID - 4 Bytes
- type - 4 Bytes
- encryptionAlgorithm - 3 Bytes
- lastWriteTime - 8 Bytes
- lastOperation - 6 Bytes
- lastWriteUserID - 4 Bytes

In Summe werden also zu jedem Datensatz (vom Typ *data*) 33 Bytes zusätzlicher Speicherplatz in der Tabelle *storage* des Datenspeichers benötigt. Bei *metadataRoot*-Datensätzen sind dies 41 Bytes und bei *metadata*-Datensätzen 37 Bytes.

Die verschlüsselten Session-Keys, welche in der Tabelle *keysForStorage* gespeichert werden, verursachen darüber hinaus noch folgenden Speicherverbrauch:

- ID - 4 Bytes
- keyID - 4 Bytes
- encryptedSessionKey - 128 Bytes
- encryptionAlgorithm - 3 Bytes

Es werden also pro verschlüsselte Session-Key 139 Bytes Speicher benötigt.

6.4.1 Speicherbedarf im Datenspeicher bei nicht historisierter Speicherung

Bei Metadaten der folglich beschriebenen Datensätze werden, wenn ein Datensatz im MetataRoot-Datensatz eingetragen wird, nur jene Daten berücksichtigt, um welche dieser Datensatz vergrößert wird. Ein Metadatensatz, in welchen ein neuer Datensatz eingetragen wird, wird dadurch um ca. 650 Bytes vergrößert. Bei Datensätzen, welche einen eigenen Metadatensatz besitzen (z.B. Zählerdatensätze), wird sowohl der ganze Metadatensatz selbst, als auch die Vergrößerung des Metadatensatzes, welcher auf den neuen Metadatensatz verweist, berücksichtigt. Diese Vergrößerung beträgt, je nach Länge des *Tags*, ca. 100 Bytes.

Leerer MetadataRoot-Datensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz	164 Bytes
zusätzliche Daten	41 Bytes
2 verschlüsselte Session-Keys	278 Bytes
Summe	**483 Bytes**

Kundeninformationsdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Anrede: 8 Zeichen Vorname: 10 Zeichen Nachname: 15 Zeichen Email-Adresse: 40 Zeichen Straße, Hausnummer: 35 Zeichen PLZ, Ort: 20 Zeichen	448 Bytes
zusätzliche Daten	33 Bytes
2 verschlüsselte Session-Keys	278 Bytes
Vergrößerung des MetadataRoot-Datensatzes	650 Bytes
Summe	**1409 Bytes**

Dokumentendatensatz

Bei Dokumentendatensätzen macht es keinen Unterschied im Speicherbedarf, ob es sich um ein Kundendokument oder ein Immobiliendokument handelt.

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name des Dokuments: 20 Zeichen Kommentar: 0 Zeichen	288 Bytes + Größe des Dokuments
zusätzliche Daten	33 Bytes
2 verschlüsselte Session-Keys	278 Bytes
Vergrößerung des MetadataRoot-Datensatzes	650 Bytes
Summe	**1249 Bytes + Größe des Dokuments**

Immobiliendatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name der Immobilie: 20 Zeichen GPS Koordinaten: 20 Zeichen Straße, Hausnummer: 35 Zeichen PLZ, Ort: 20 Zeichen Art der Nutzung: 10 Zeichen Bauweise: 20 Zeichen Grundfläche: 8 Zeichen Nutzfläche: 8 Zeichen Anzahl der Geschoße: 1 Zeichen Art der Wärmedämmung: 25 Zeichen Kommentar: 0 Zeichen	656 Bytes
zusätzliche Daten	33 Bytes
2 verschlüsselte Session-Keys	278 Bytes
Vergrößerung des MetadataRoot-Datensatzes	650 Bytes
Summe	**1617 Bytes**

Zählerdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name des Zählers: 20 Zeichen Art des Zählers: 10 Zeichen automatisch/manuell: 9 Zeichen Messwerteinheit: 5 Zeichen Zähler-ID: 20 Zeichen Kommentar: 0 Zeichen	352 Bytes
zusätzliche Daten	33 Bytes
1 verschlüsselter Session-Key	139 Bytes
Vergrößerung des MetadataRoot-Datensatzes durch den Verweis auf den neuen Metadata-Datensatz	100 Bytes
serialisierter und verschlüsselter Metadata-Datensatz des Zählers	572 Bytes
zusätzliche Daten des Metadata-Datensatzes	37 Bytes
2 verschlüsselte Session-Keys für den Metadata-Datensatz	278 Bytes
Summe	**1620 Bytes**

Parameterdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz:	3840 Bytes
1 verschlüsselter Session-Key	139 Bytes
Vergrößerung des Metadata-Datensatzes des Zählers durch den Verweis auf den neuen Metadata-Datensatz	100 Bytes
serialisierter und verschlüsselter Metadata-Datensatz des Parameterdatensatzes	528 Bytes
zusätzliche Daten des Metadata-Datensatzes	37 Bytes
2 verschlüsselte Session-Keys für den Metadata-Datensatz	278 Bytes
Summe	**4922 Bytes**

Messwertdatensätze

Hier werden die Messwertdatensätze eines ganzen Jahres berechnet, wobei davon ausgegangen wird, dass ein automatischer Zähler alle 15 Minuten (wie beim *Google PowerMeter*) einen neuen Messwertdatensatz liefert.

	Speicherverbrauch
4*24*365=35040 mal - serialisierte und verschlüsselte Messwertdatensätze	35040*304 Bytes = 10652160 Bytes
4*24*365=35040 mal - zusätzliche Daten zu den Messwertdatensätzen	35040*33 Bytes = 1156320 Bytes
4*24*365*2=70080 verschlüsselte Session-Keys	70080*139 Bytes = 9741120 Bytes
Vergrößerung des Metadata-Datensatzes des Zählers durch den Verweis auf den neuen Metadata-Datensatz des Jahres	100 Bytes
serialisierter und verschlüsselter Metadata-Datensatz des Jahres (leer)	164 Bytes
zusätzliche Daten zum Metadata-Datensatz des Jahres	37 Bytes
2 verschlüsselte Session-Keys für den Metadata-Datensatz des Jahres	278 Bytes
52 serialisierte und verschlüsselte Metadata-Datensätze für die Kalenderwochen des Jahres (leer)	52*164 Bytes = 8528 Bytes
52 mal - zusätzliche Daten zu den Metadata-Datensätze der Kalenderwochen	52*37 Bytes = 1924 Bytes
2*52=104 verschlüsselte Session-Keys für die Metadata-Datensätze der Kalenderwochen	104*139 Bytes = 14456 Bytes
52 mal - Vergrößerung des Metadata-Datensatzes des Jahres durch den Verweis auf die Metadata-Datensätze der Kalenderwochen	52*100 Bytes = 5200 Bytes

	Speicherverbrauch
4*24*365=35040 mal - Vergrößerung der Metadata-Datensätze der entsprechenden Kalenderwoche durch die Eintragung der Messwertdatensätze	35040*624 Bytes = 21864960 Bytes
Summe	**43335247 Bytes = 41,43 Megabyte**

Dies bedeutet, dass ein automatischer Zähler, welcher alle 15 Minuten Daten einliefert, jährlich im Datenspeicher ein Datenaufkommen von 41,43 MB verursacht. Dabei wird rund ein Viertel durch die verschlüsselten Messwertdatensätze selbst, ein Viertel durch die verschlüsselten Session-Keys und die restliche Hälfte durch die Metadaten eingenommen.

Bei manuellen Zählern ist das Datenaufkommen natürlich entsprechend der Ablesehäufigkeit weitaus kleiner.

Fazit

Man erkennt deutlich, dass ein Großteil des Speichers durch die Messwerte und deren Metadaten eingenommen wird. Einen weiteren großen Faktor stellen die Dokumente dar, welche auch bis zu einigen Megabyte groß sein können. Die Immobilien-, Kundeninformations-, Zähler- und Parameterdatensätze fallen kaum ins Gewicht.

6.4.2 Speicherbedarf im Datenspeicher bei historisierter Speicherung

Kundeninformations-, Dokumenten-, Immobilien-, Zähler- und Parameterdatensätze unterscheiden sich, sofern diese nicht editiert werden, nicht von jenen der nicht historisierten Datenspeicherung. Lediglich die zusätzlichen Datenfelder in der Datenbank, welche für die Historisierung gebraucht werden, müssen berücksichtigt werden. Dies sind pro Datensatz 13 Bytes (4 Bytes für die Versionsnummer, 8 Bytes für den Löschzeitpunkt und 1 Byte für das Löschflag). Außerdem wird der Metadatensatz, in welchen der entsprechende Datensatz eingetragen wird, nicht vergrößert, sondern neu gespeichert.
Beispiel: Angenommen ein neuer Datensatz wird in einem Metadata-Datensatz, welcher vorher eine Größe von 1248 Bytes Rohdaten + 50 Bytes zusätzliche Daten + 278 Bytes für 2 verschlüsselte Session-Keys = 1576 Bytes hat, eingetragen. Angenommen die Metadaten des neuen Datensatzes würden den Metadatensatz um 650 Bytes vergrößern. Durch die Historisierung würde der benötigte Speicher für die Metadaten danach 1576 Bytes + 1248 Bytes + 650 Bytes + 50 Bytes zusätzliche Daten + 278 Bytes für 2 verschlüsselte Session-Keys = 3802 Bytes betragen. D.h., dass der Speicherverbrauch für die Metadaten extrem ansteigt. Bei der nicht historisierten Speicherung wären dafür lediglich 2213 Bytes notwendig gewesen. Dieser Effekt verstärkt sich bei jedem Datensatz, welcher hinzukommt. Abbildung 6.3 führt das Beispiel bis zum 15. neuen Datensatz fort.

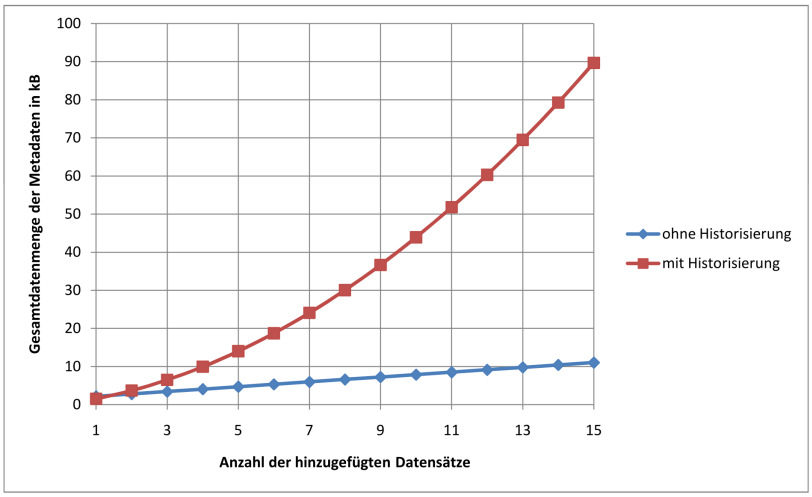


Abbildung 6.3: Anstieg der Datenmenge der Metadaten beim Hinzufügen von neuen Datensätzen

Kundeninformationsdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Anrede: 8 Zeichen Vorname: 10 Zeichen Nachname: 15 Zeichen Email-Adresse: 40 Zeichen Straße, Hausnummer: 35 Zeichen PLZ, Ort: 20 Zeichen	448 Bytes
zusätzliche Daten	46 Bytes
2 verschlüsselte Session-Keys	278 Bytes
aktualisierter MetadataRoot-Datensatz	650 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen
zusätzliche Daten für den MetadataRoot-Datensatz	54 Bytes
2 verschlüsselte Session-Keys für den MetadataRoot-Datensatz	278 Bytes
Summe	**1754 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen**

Dokumentendatensatz

Bei Dokumentendatensätzen macht es keinen Unterschied im Speicherbedarf, ob es sich um ein Kundendokument oder ein Immobiliendokument handelt.

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name des Dokuments: 20 Zeichen Kommentar: 0 Zeichen	288 Bytes + Größe des Dokuments
zusätzliche Daten	46 Bytes
2 verschlüsselte Session-Keys	278 Bytes
aktualisierter MetadataRoot-Datensatz	650 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen
zusätzliche Daten für den MetadataRoot-Datensatz	54 Bytes
2 verschlüsselte Session-Keys für den MetadataRoot-Datensatz	278 Bytes
Summe	**1594 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen + Größe des Dokuments**

Immobiliendatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name der Immobilie: 20 Zeichen GPS Koordinaten: 20 Zeichen Straße, Hausnummer: 35 Zeichen PLZ, Ort: 20 Zeichen Art der Nutzung: 10 Zeichen Bauweise: 20 Zeichen Grundfläche: 8 Zeichen Nutzfläche: 8 Zeichen Anzahl der Geschoße: 1 Zeichen Art der Wärmedämmung: 25 Zeichen Kommentar: 0 Zeichen	656 Bytes
zusätzliche Daten	46 Bytes
2 verschlüsselte Session-Keys	278 Bytes

	Speicherverbrauch
aktualisierter MetadataRoot-Datensatz	650 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen
zusätzliche Daten für den MetadataRoot-Datensatz	54 Bytes
2 verschlüsselte Session-Keys für den MetadataRoot-Datensatz	278 Bytes
Summe	**1962 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen**

Zählerdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz: Name des Zählers: 20 Zeichen Art des Zählers: 10 Zeichen automatisch/manuell: 9 Zeichen Messwerteinheit: 5 Zeichen Zähler-ID: 20 Zeichen Kommentar: 0 Zeichen	352 Bytes
zusätzliche Daten	46 Bytes
1 verschlüsselter Session-Key	139 Bytes
aktualisierter MetadataRoot-Datensatz (Hinzufügen eines Verweises auf den neuen Metadata-Datensatz des Zählers)	100 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen
zusätzliche Daten für den MetadataRoot-Datensatz	54 Bytes
2 verschlüsselte Session-Keys für den MetadataRoot-Datensatz	278 Bytes
serialisierter und verschlüsselter Metadata-Datensatz des Zählers	572 Bytes
zusätzliche Daten des Metadata-Datensatzes	50 Bytes
2 verschlüsselte Session-Keys für den Metadata-Datensatz	278 Bytes
Summe	**1869 Bytes + Größe des verschlüsselten MetadataRoot-Datensatzes vor dem Hinzufügen**

Parameterdatensatz

	Speicherverbrauch
serialisierter und verschlüsselter Datensatz:	3840 Bytes
1 verschlüsselter Session-Key	139 Bytes
aktualisierter Metadatensatz des Zählers (Hinzufügen eines Verweises auf den neuen Metadata-Datensatz des Parameterdatensatzes)	100 Bytes + Größe des verschlüsselten Metadata-Datensatzes des Zählers vor dem Hinzufügen
serialisierter und verschlüsselter Metadata-Datensatz des Parameterdatensatzes	528 Bytes
zusätzliche Daten des Metadata-Datensatzes	50 Bytes
2 verschlüsselte Session-Keys für den Metadata-Datensatz	278 Bytes
Summe	**4935 Bytes + Größe des verschlüsselten Metadata-Datensatzes des Zählers vor dem Hinzufügen**

Messwertdatensätze

Wie bei der nicht historisierten Speicherung, werden die Messwertdatensätzes eines ganzen Jahres berechnet, wobei davon ausgegangen wird, dass ein automatischer Zähler alle 15 Minuten einen neuen Messwertdatensatz liefert. Da sich ein kleiner Fehler bei der Schätzung der Größe der serialisierten und verschlüsselten Metadata-Datensätze verstärkt fortpflanzen würde, wurden diese mit Hilfe der Simulation einer Woche durch die Demo-Applikation ermittelt.

	Speicherverbrauch
4*24*365=35040 mal - serialisierte und verschlüsselte Messwertdatensätze	35040*304 Bytes = 10652160 Bytes
4*24*365=35040 mal - zusätzliche Daten zu den Messwertdatensätzen	35040*46 Bytes = 1611840 Bytes
4*24*365*2=70080 verschlüsselte Session-Keys	70080*139 Bytes = 9741120 Bytes
aktualisierter Metadata-Datensatz (Hinzufügen eines Verweises auf den Metadata-Datensatz des Jahres)	100 Bytes + Größe des verschlüsselten Metadata-Datensatzes des Zählers vor dem Hinzufügen des Metadata-Datensatzes des Jahres
zusätzliche Daten des Metadata-Datensatzes des Zählers	50 Bytes

	Speicherverbrauch
2 verschlüsselte Session-Keys für den Metadata-Datensatz des Zählers	278 Bytes
serialisierter und verschlüsselter Metadata-Datensatz des Jahres (52 mal aktualisiert; Vergrößerung um 100 Bytes bei jeder Aktualisierung)	146328 Bytes
52 mal - zusätzliche Daten für jede Aktualisierung des Metadata-Datensatzes des Jahres	52*50 Bytes = 2600 Bytes
52*2=104 verschlüsselte Session-Keys für den Metadata-Datensatz des Jahres (je 2 pro Aktualisierung)	104*139 Bytes = 14456 Bytes
52 serialisierte und verschlüsselte Metadata-Datensätze für die Kalenderwochen des Jahres (jeder 672 mal aktualisiert)	52*142852416 Bytes = 6,92 Gigabyte
52*672=34944 mal - zusätzliche Daten zu den Metadata-Datensätzen der Kalenderwochen (inkl. Aktualisierungen)	52*672*50 Bytes = 1747200 Bytes
2*52*672=69888 verschlüsselte Session-Keys für die Metadata-Datensätze der Kalenderwochen (inkl. Aktualisierungen)	2*52*672*139 Bytes = 9714432 Bytes
Summe	**7461956196 Bytes = 6,95 Gigabyte + Größe des verschlüsselten Metadata-Datensatzes des Zählers vor dem Hinzufügen des Metadata-Datensatzes des Jahres**

Dies bedeutet, dass ein Zähler, welcher ein Jahr lang alle 15 Minuten einen Messwertdatensatz einbringt, Speicherplatz von fast 7 GB in der Datenbank belegt, wenn die Speicherung historisiert stattfindet. Man erkennt deutlich, dass der Speicherverbrauch fast zur Gänze von den Metadaten verursacht wird. Außerdem wird bei historisierter Speicherung ungefähr 170 mal mehr Speicher benötigt, als bei nicht historisierter Speicherung.

Fazit

Auch bei der historisierten Speicherung wird nahezu der ganze verbrauchte Speicher für die Speicherung der Messwertdatensätze und vor allem für die Metadaten dieser eingenommen. Dokumente, welche einige Megabyte groß sind, fallen hier nicht mehr stark ins Gewicht. Es wird sehr viel mehr Speicher benötigt als bei der nicht historisierten Speicherung, was die Performance der Datenbank verschlechtert und darüber hinaus erhebliche Mehrkosten verursacht. Der erhöhte Speicherverbrauch ist vor allem durch die vielen Updates der Metadatensätze zu erklären.

6.4.3 Speicherbedarf in der Schlüsselverwaltung

Jedes Schlüsselpaar benötigt zur Speicherung in der Datenbank ungefähr 1200 Byte. Dies bedeutet, dass für jeden Kundeninformations-, Dokumenten- und Immobiliendatensatz Speicher von 1200 Bytes in der Schlüsselverwaltung belegt wird. Pro Immobilie wird ein Schlüsselband angelegt, welches pro Jahr bis zu 52 Schlüssel haben kann. Dies ergibt pro Immobilie jährlich 62400 Byte. Pro automatischem Zähler wird außerdem ein Signaturschlüssel angelegt, welcher auch 1200 Byte belegt.

Pro freigegebenem oder angefordertem Schlüssel werden zusätzlich 37 Bytes belegt.

Anhand dieser Zahlen ist leicht zu erkennen, dass die Daten, welche in der Schlüsselverwaltungsdatenbank abgelegt sind, bei Weitem nicht die Größe jener im Datenspeicher erreichen.

6.5 Zusammenfassung

In diesem Kapitel wurden einige wichtige Punkte des Designkonzepts diskutiert. Dazu wurde anfangs die Konsistenzhaltung der Daten eines Kunden im Datenspeicher analysiert. Dabei wurde festgestellt, dass neben der Sicherung des exklusiven Schreibzugriffs auf den Datenspeicher durch den Lock-Mechanismus auch die korrekte Implementierung des Clients von essenzieller Bedeutung für die Konsistenz der Daten ist. Außerdem wurde analysiert, was hinsichtlich der Konsistenz der Kundendaten passiert, wenn ein Fehler während eines Systemprozesses auftritt.

Im darauffolgenden Abschnitt wurde am Beispiel eines Systemprozesses ein Komplexitätsvergleich zwischen verschlüsselten und unverschlüsselten Metadaten angestellt. Dabei wurde festgestellt, dass die Komplexität im Fall von verschlüsselten Metadaten erheblich höher ist.

Im nächsten Abschnitt wurde ein Verbesserungsvorschlag für den Lock-Mechanismus vorgestellt, welcher die Parallelität von Schreiboperationen auf den Datenspeicher wesentlich erhöhen sollte.

Abschließend wurde der Speicherverbrauch in der Datenbank durch die Speicherung der Kundendaten bei historisierter und nicht historisierter Speicherung analysiert. Dabei wurde festgestellt, dass die historisierte Speicherung der Messdaten bei einem Messintervall von 15 Minuten 170 mal so hoch ist wie bei nicht historisierter Speicherung.

KAPITEL 7

Vergleich mit dem Google PowerMeter

Der Suchmaschinenanbieter *Google* hat vor kurzem ein Projekt namens „Google PowerMeter“ gestartet. Dieses System soll es ermöglichen, den Stromverbrauch eines Haushalts in Echtzeit im Internet über ein *iGoogle*-Wigdet anzuzeigen. Dieses Widget zeigt eine Kurve mit dem Verlauf des Stromverbrauchs über die Zeit (siehe Abbildung 7.1). Außerdem bietet das Widget einen Vergleich des Tagesverbrauchs mit Standardwerten bestimmter Größen von Wohnungen und Häusern. Zusätzlich kann der Benutzer seinen aktuellen Stromverbrauch mit dem Verbrauch vergangener Tage/Wochen/Monate vergleichen um somit Verbesserungen oder Verschlechterungen der Verbrauchswerte besser beobachten zu können.

Als Grund für die Entwicklung des Systems gibt *Google* an, den Stromverbrauch weltweit zu senken und somit einerseits die Stromkosten für die Verbraucher zu senken und andererseits die Notwendigkeit nach neuen Kraftwerken zu reduzieren. Laut *Google* haben Studien gezeigt, dass durch die bloße Anzeige des Stromverbrauchs in Echtzeit der Energiebedarf in Haushalten zwischen 5% und 15% gesenkt werden kann [2]. Wenn der Kunde zusätzlich noch Informationen und Tipps zur Senkung des Energiebedarfs erhält, könne der Einsparungseffekt noch beträchtlich größer sein. Als weiteren Grund für die Entwicklung von *Google PowerMeter* gibt *Google* die Senkung des CO2 Ausstoßes an. Somit könne jeder mit Hilfe des Systems seinen persönlichen CO2-Footprint erheblich verkleinern.

Zur Zeit befindet sich das System in der Entwicklungs- bzw. Testphase. Außerdem versucht *Google* momentan in möglichst vielen Ländern Partnerfirmen zu finden, welche zum einen Geräte, also kompatible Stromzähler, auf den Markt bringen und zum anderen Energieversorger, welche diese anschließend vertreiben. So gibt es in einigen Ländern schon Testprojekte, welche *Google PowerMeter* benutzen. Unter diesen Kooperationspartnern befinden sich mittlerweile auch einige europäische Energieversorger. In Großbritannien bietet der Energieversorger „First Utility“, welcher seit Septem-

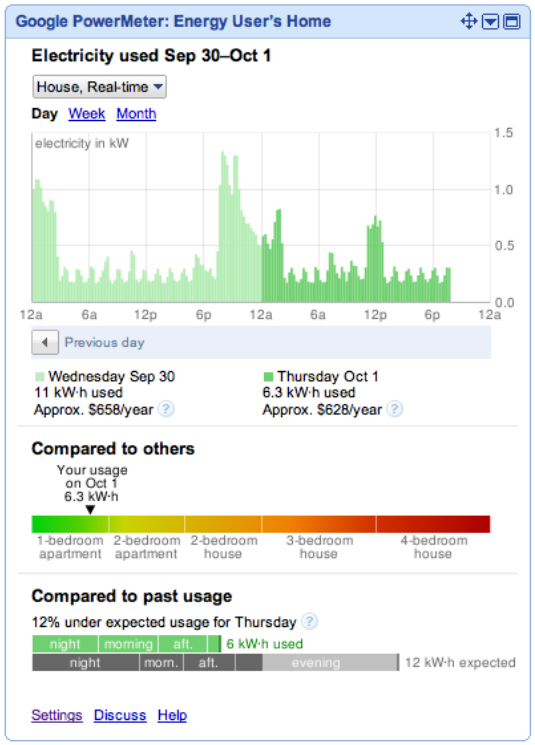


Abbildung 7.1: Screenshot des Google PowerMeter Widgets [16]

ber 2008 30000 Kunden mit Smart Meters ausgestattet hat, die Möglichkeit, die Messdaten in das *Google PowerMeter* einzuspeisen. Außerdem bietet die Firma „AlertMe“ ein System, mit welchem sich ein drahtloses *Home Area Network (HAN)* aufbauen lässt, welches auf *Zigbee* beruht und durch den Kunden selbst installiert werden kann. Dieses System bietet unter anderem die Möglichkeit, entweder den Verbrauch einzelner Steckdosen durch Zwischenstecker zu messen oder durch einen Aufsatz auf den normalen Stromzähler den Verbrauch des ganzen Hauses zu messen. Diese Messdaten werden dann an einen Hub geschickt, welcher mit dem Internet verbunden ist und die Messdaten anschließend an das *Google PowerMeter* schickt [15]. Der Vorteil dieses Systems ist neben der selbstständigen Montage, dass der Energieversorger umgangen wird.

In Deutschland bietet zur Zeit der Stromanbieter „Yello Strom“ den sogenannten „Sparzähler“ an, welcher ebenfalls kompatibel mit dem *Google PowerMeter* ist und alle 15 Minuten aktuelle Messdaten an *Google* sendet.

Allerdings gibt es auch Kritik am *Google PowerMeter*. So fordert beispielsweise Jim Stogdill in einem Blogeintrag [20] unter anderem die Freigabe der API für das PowerMeter, damit jeder, der sich beteiligen möchte, eigene Geräte anbieten kann. Dies ist zwar bis jetzt noch nicht geschehen, jedoch erklärt *Google* die Absicht, dies



Abbildung 7.2: Der Sparzähler von Yello Strom [26]

zukünftig zu tun [2].

7.1 Vergleich: Google PowerMeter vs Open Facility Data Portal

Die grundsätzliche Idee der beiden Systeme ist sehr ähnlich, allerdings bietet das hier gezeigte *Open Facility Data Portal* in einigen Bereichen wesentlich mehr. So ist es beispielsweise mit dem *Open Facility Data Portal* möglich, Messdaten aller Art einzuspeisen, also auch Gasverbrauch, Fernwärmeverbrauch und andere Verbrauchsdaten, wohingegen das *Google PowerMeter* lediglich die Möglichkeit bietet, den Stromverbrauch einzuspeisen und zu visualisieren. Außerdem bietet das *Open Facility Data Portal* die Möglichkeit der kontrollierten Weitergabe der Messdaten an Energieberater. Beim *Google PowerMeter* gibt es diese Möglichkeit bislang nicht.

Ein weiterer Vorteil des *Open Facility Data Portal*s ist, dass jeder Kunde frei wählen kann, welchem Hosting Provider er seine Daten zur Speicherung anvertraut. Darüber hinaus werden die Daten verschlüsselt gespeichert, wodurch der Hosting Provider die Daten in keiner Form verwerten kann. Beim *Google PowerMeter* ist die Speicherung lediglich bei *Google* selbst möglich. Da bislang nicht offengelegt wird, wie die Speicherung erfolgt bzw. keine Wahlfreiheit des Speicheranbieters besteht, muss der Benutzer *Google* vertrauen, dass seine Daten vertraulich behandelt werden.

Weiters ist anzumerken, dass das *Google PowerMeter* gegenüber des *Open Facility Data Portal*s nicht die Möglichkeit von Profilen zu einzelnen Immobilien bietet. Auch der fehlende Dokumentenupload zu den entsprechenden Immobilien ist zu erwähnen.

Zusammenfasssend lässt sich sagen, dass der wesentliche Unterschied zwischen dem *Open Facility Data Portal* und dem *Google PowerMeter* darin besteht, dass das *Open Facility Data Portal* mehr auf die Energieberatung abzielt als auf die bloße Anzeige des eigenen Stromverbrauchs, wie das beim *Google PowerMeter* der Fall ist. Darüber hinaus ist das Thema des Datenschutzes beim *Google PowerMeter* eher fraglich, da nicht offengelegt wird, wie die Daten gespeichert werden.

Funktion	**Google PowerMeter**	**Open Facility Data Portal**
automatische Messdatenerfassung	+	+
händische Messdatenerfassung	-	+
Erfassung von Strommessdaten	+	+
Erfassung anderer Messdaten (Gas, Fernwärme,...)	-	+
verschlüsselte Speicherung der Daten	-	+
freie Wahl des Speicheranbieters	-	+
Anzeige der Messdaten in Echtzeit	+	+
Anlegen von Profilen zu Immobilien	-	+
Upload von Dokumenten zu den Immobilien	-	+
Kontrollierte Weitergabe der Daten an den Energieberater	-	+
Erfahrungsaustausch der Energiekunden	-	+

KAPITEL 8

Zusammenfassung und Ausblick

Zum Abschluss der Arbeit sollen nun die Ergebnisse noch einmal kurz zusammengefasst werden. Außerdem werden Anregungen und Verbesserungsvorschläge zur weiteren Entwicklung des Systems gegeben.

Zu Beginn der Arbeit wurde ein kurzer Einblick in den Stand der Technik in den Themenbereichen *Smart Metering*, *Verschlüsselung von Daten* und *Web Services* gegeben. Anschließend wurde die Aufgabenstellung analysiert und erste Ideen zur Lösung des Problems dargelegt. Dazu wurden schon erste grobe Abläufe vorgestellt und Komponenten identifiziert. Abgeschlossen wurde diese Analyse durch die genaue Spezifikation der Anforderungen an das System.

Im nachfolgenden Kapitel wurde das Designkonzept vorgestellt. Dazu wurden zuerst die Daten, welche ein Kunde im System speichern kann, beschrieben und anschließend wurde auf deren Verschlüsselung eingegangen. Hiernach wurde die Verschlüsselung der Metadaten diskutiert und der dazu notwendige Lock-Mechanismus vorgestellt. Anschließend wurden die Aufgabenbereiche der in der Analyse identifizierten Komponenten detailliert beschrieben, welche hier noch einmal kurz zusammenfasst werden. Das System umfasst folgende Komponenten: die Schlüsselverwaltung, ein oder mehrere Datenspeicher, die automatischen Datenquellen und die Clients der Kunden und der Energieberater, wobei diese in einer realen Implementierung in ein Web-Portal mit Social-Network Charakter eingebunden sind.

Die Schlüsselverwaltung ist für die Erzeugung, Speicherung und die Verwaltung der Zugriffsrechte der kryptographischen Schlüssel und Signaturschlüssel der Kunden verantwortlich. Außerdem werden in der Schlüsselverwaltung Schlüsselanfragen von Beratern gespeichert.

Ein Kunde kann sein Datenaufkommen in einem oder mehreren Datenspeichern ablegen, wobei alle Daten, welche zu einer Immobilie gehören, im selben Datenspeicher gespeichert sein müssen. Alle in den Datenspeichern abgelegten Daten und Meta-

daten werden ausschließlich verschlüsselt gespeichert. Nur der Kunde selbst oder von ihm ausgewählte Berater können diese Daten entschlüsseln, sofern der Kunde diesen Beratern den Zugriff auf die entsprechenden Schlüssel erteilt hat. Bei der Ver- und Entschlüsselung der Datensätze kommt ein hybrides Verschlüsselungsverfahren zum Einsatz, welches es ermöglicht, dass der selbe Datensatz mit mehreren Schlüsseln gleichzeitig verschlüsselt ist, ohne die Daten selbst mehrmals zu verschlüsseln (es wird jeweils nur der Session-Key neu verschlüsselt).

Die automatischen Datenquellen (automatische Zähler) können ohne Zutun des Kunden automatisch Messwerte in das System einbringen. Alle Requests einer automatischen Datenquelle werden digital signiert. Wenn die Schlüsselverwaltung oder ein Datenspeicher einen solchen Request empfängt, muss zuerst die Signatur dieses Requests auf Gültigkeit überprüft werden. Die Signatur sichert die Integrität des Requests und authentifiziert Datenquelle.

Die Clients der Kunden und der Berater sind für die Ver- und Entschlüsselung der Datensätze und für die Darstellung der Daten verantwortlich. Aufgrund der Verschlüsselung der Daten und vor allem aufgrund der Verschlüsselung der Metadaten, muss der Kundenclient einen großen Teil zur Konsistenzhaltung der Daten beitragen (siehe Abschnitt 4.6 und Abschnitt 6.1).

Die Weitergabe der Daten eines Kunden an einen Energieberater erfolgt durch die Gewährung des Zugriffs auf die Private-Keys der Schlüsselpaare der entsprechenden Datensätze.

Nachdem alle allgemeinen Konzepte des Designs vorgestellt wurden, konnten sehr detaillierte Systemprozesse und die Schnittstellen zu den einzelnen Komponenten entwickelt werden. Das Designkonzept wurde abgeschlossen, indem eine Methode entwickelt wurde, welche es erlaubt, die Daten eines Kunden in historisierter Form zu speichern. Dadurch kann der Datenbestand eines Kunden zu einem beliebigen Zeitpunkt in der Vergangenheit wiederhergestellt werden. Als Beweis für die Korrektheit und Vollständigkeit des Konzepts wurde anschließend eine Demo-Applikation implementiert, welche alle Systemprozesse und alle Schnittstellenfunktionen implementiert. Im Zuge der Beschreibung der Demo-Applikation wurde gezeigt, wie die Daten eines Kunden in einer SQL-Datenbank gespeichert werden können.

Im Anschluss an die Beschreibung der Demo-Applikation wurde das entwickelte Designkonzept eingehend diskutiert. Dabei wurde unter anderem festgestellt, dass die Komplexität der Systemprozesse durch die Verschlüsselung der Metadaten wesentlich höher ist, als jene von unverschlüsselten Metadaten. Es wurde ebenfalls festgestellt, dass der vorgestellte Lock-Mechanismus die Parallelität von Schreibzugriffen auf die Datenspeicher einschränkt. Dies kann allerdings durch Implementierung eines etwas komplizierten Mechanismusses erheblich verbessert werden. Weiters wurde der Speicherbedarf der Datenbank durch die Speicherung der Kundendaten analysiert. Dabei stellte sich heraus, dass für die Speicherung von Energiemessdaten die Implementierung mit historisierter Speicherung ungefähr den 170-fachen Speicherbedarf hat, wie jene im nicht historisierten Fall.

Abschließend wurde das vorliegende System mit dem *Google PowerMeter* verglichen, wobei dieses bei weitem nicht die Möglichkeiten des vorliegenden Systems aufweist. Vor allem bietet das *Google PowerMeter* nicht die Möglichkeiten der verschlüsselten Speicherung der Daten und der kontrollierten Weitergabe dieser.

Welche Möglichkeiten der Weiterentwicklung bietet das vorgestellte System? Es hat sich herausgestellt, dass die Entschlüsselung von sehr vielen kleinen Datensätzen, wie das bei den Messwertdatensätzen der Fall ist, mitunter sehr viel Zeit in Anspruch nehmen kann. Daher wäre es sinnvoll, die Messdaten wochenweise im Nachhinein zu einem einzigen Datensatz zusammenzufassen. Allerdings wirft diese Vorgangsweise einige Fragen auf: Was passiert, wenn später neue Messdaten dieser Woche hinzugekommen? Dies könnte bei händischen Zählern durchaus der Fall sein. Wer stößt diesen Prozess an? Dies müsste eigentlich der Kunde selbst sein, da nur dieser die nötigen Berechtigungen aufweist.

Weiters wäre es vorstellbar, einen Kommunikationskanal zu weiteren Geräten zu schaffen. So könnte beispielsweise eine Waschmaschine so gesteuert sein, dass sie sich dann einschaltet, wenn der Strom am günstigsten ist (eine der Ideen eines *Smart Grids*). Ein weiteres Beispiel wäre eine Heizungssteuerung, welche der Besitzer über das Internet steuern könnte. Diese Funktionalität könnte so implementiert werden, dass in jedem Gerät ein gerätespezifischer Private-Key, die Adresse eines Datenspeichers und eine ID eines Datensatzes abgelegt sind. Die entsprechenden Daten sind dann bei diesem Datenspeicher unter dieser ID abzuholen und nur mit dem Private-Key des Geräts lesbar. Das Gerät fragt nun in regelmäßigen Abständen diesen Datensatz ab. Somit könnte über diesen Datensatz eine Steuerung dieses Geräts stattfinden, wobei weiterhin alle Daten im Datenspeicher nur in verschlüsselter Form vorliegen. Allerdings stellt sich hierbei eine andere, eher organisatorische Frage: Wer erzeugt im Beispiel mit der Waschmaschine den Datensatz, welcher die Waschmaschine schlussendlich einschaltet?

Man erkennt, dass das Potential zur Weiterentwicklung des Systems noch keinesfalls erschöpft ist. Diese Arbeit bietet die Grundlage für weitere Entwicklungen in diesem Themenbereich. Außerdem kann die Arbeit direkt für die reale Umsetzung des beschriebenen Systems herangezogen werden.

ANHANG A

Anhang

A.1 Stichwortverzeichnis

Backup-Schlüssel - Dieses Schlüsselpaar wird direkt nach der Registrierung des Kunden erzeugt. Mit dem Private-Key dieses Schlüsselpaares ist es möglich jeden Datensatz eines Kunden zu entschlüsseln. Dieser Schlüssel hat die Funktion eines Notfallschlüssels für den Fall, dass andere Schlüssel (z.B. Wochenschlüssel) des Kunden verloren gehen. Außerdem kann der Schlüssel als Universalschlüssel für alle Datensätze des Kunden verwendet werden. Um das zu ermöglichen, muss die Datenquelle bei der automatischen Messwerterfassung (z.B. Zähler), neben dem aktuellen Wochenschlüssel auch den Public Key des Backup-Schlüssels zur Verschlüsselung verwenden.

Benutzer - Ein Benutzer ist ein User des Systems. Es kann sich dabei sowohl um einen Kunden, als auch um einen Energieberater handeln. Dieser Begriff wird immer dann verwendet, wenn keine Unterscheidung zwischen Kunden und Beratern getroffen wird.

Datenspeicher - Jene Komponeten des Systems, bei welcher ein Kunde seine Daten speichert. Der Kunde, kann seine Daten auf mehere Datenspeicher verteilen.

Dokumentenschlüssel - siehe **individuelle Datensatzschlüssel**

Hosting Provider - siehe **Datenspeicher**

individuelle Datensatzschlüssel - Dieses Schlüsselpaar ist einem einzigen Datensatz eines Kunden zugeordnet. Dies hat den Vorteil, dass es bei der Freigabe möglich ist, nur genau diesen Datensatz für einen Berater freizugeben (durch Gewähren des Zugriffs auf den Private-Key dieses Schlüsselpaares). Diese Schlüsselard wird bei folgen-

den Datensätzen eingesetzt: Kunden-Informationsdatensatz, Immobilien-Informationsdatensatz und Dokumente.

Immobilie - Jeder Kunde kann mehrere Immobilien anlegen. Dabei bildet jede Immoblilie den Wurzelknoten der darunterliegenden Daten (in der Baumstruktur der Kundendaten). Die Immobilie selbst wird durch den Immobilien-Informationsdatensatz beschrieben. In diesem Datensatz sind deskriptive Daten der Immobilie gespeichert (Adresse, GPS-Koordinaten, Bauweise, Art der Nutzung, Art der Heizung, Grundfläche, Nutzfläche,...). Zu jeder Immobilie können eine beliebige Zahl von Zähler inventarisiert werden, wobei für die Verschlüsselung/Entschlüsselung der Messdaten aller Zähler einer Immobilie die gleichen Wochenschlüssel verwendet werden. Dies hat zur Folge, dass wenn ein Kunde Messdaten für eine Immobilie für einen Berater freigibt (d.h. den Zugriff auf die entsprechenden Wochenschlüssel gewährt), dieser Berater gleichzeitig Zugriff auf alle Messdaten dieser Immobilie erhält. Freigaben von Daten nur einzelner Zähler sind nicht vorgesehen. Zu jeder Immobilie können auch Dokumente hochgeladen und freigegeben werden. Hier könnten bespielsweise Bilder der Immobilie, den Energieausweis, Wärmebildmessungen, letzte Energieabrechnungen gespeichert werden.

Immobiliendokumente - Bei dieser Art von Dokumenten handelt es sich um Dokumentendatensätze, welche einer bestimmten Immobilie eines Kunden zugeordnet sind. Es ist vorgesehen dass ein Kunde Dokumenten wie den Energieausweis einer Immobilie, die letzten Energieabrechnungen, Wärmebildmessungen als Immobiliendokumente abspeichert.

Inventarisierung - Dies ist jener Vorgang, bei welchem dem System ein neuer Zähler/ eine neue Datenquelle für die Messdatenerfassung bekannt gemacht wird. Dabei wird zuerst vom Kunden die zugehörige Immobilie ausgewählt, anschließend die deskriptiven Daten zum neuen Zähler eingegeben und danach mit dem Public-Key des Backupschlüsselpaares verschlüsselt. Danach wird der Zähler-Informationsdatensatz im Datenspeicher gespeichert. Bei automatischen Zählern wird zusätzlich ein Parameterdatensatz angelegt, welcher alle für den Betrieb des Zählers notwendigen Daten enthält (siehe Paramterdatensatz).

Kundendokumente - Bei Kundendokumenten handelt es sich um Dokumentendatensätze welche keiner speziellen Immobilie zugeordnet sind und daher in der Baumstruktur der Kundendaten direkt dem Kundenknoten zugeordnet werden.

Messdatensatz - Ein Datensatz welcher einen Messwert eines Zählers, den Zeitpunkt der Messung und einige deskriptive Metadaten (z.B. Messwerteinheit) enthält. Die Daten eines Messdatensatzes sind mit dem zum Zeitpunkt der Messung aktuellen Public-Key des in der Datenquelle parametrierten Schlüsselbandes und dem Public-Key des Backup-Schlüsselpaares verschlüsselt.

Parameterdatensatz - Dieser Datensatz enthält alle für die Parametrierung einer automatischen Datenquelle notwendigen Daten. Der Datensatz enthält zum Beispiel den

Signaturschlüssel für die Authentifizierung des Zählers, die Nummer des Schlüsselbandes, vom welchem die Wochenschlüssel für die Verschlüsselung stammen, den Public-Key des Backupschlüsselpaares mit welchem die Messdaten zusätzlich verschlüsselt werden und die Daten zum Auffinden des zugehörigen Datenspeichers und der Schlüsselvewaltung.

Signaturschlüssel der Datenquelle - In jeder Datenquelle (z.B. Zähler) ist ein Private-Key eines Signaturschlüsselpaares parametriert. Dieses dient zur Signatur aller Requests welche von der Datenquelle an den Datenspeicher oder an die Schlüsselverwaltung sendet. Diese Signatur dient zur dient zur Integritäts- und Authentizitätsprüfung durch das entsprechende Web-Service.

Wochenschlüssel - Der Public-Key des Wochenschlüsselpaares dient zur Verschlüsselung von Messdaten. D.h. ein Messwert, welcher in einer bestimmten Woche abgelesen wurde (automatisch oder manuell) wird (neben dem Backup-Schlüssel) mit dem Public-Key dieser Kalenderwoche verschlüsselt. Der Kunde kann nun mithilfe des entsprechenden Private-Keys dieses Wochenschlüssels die Messdatensätze wieder entschlüsseln. Durch Weitergabe von Private-Keys von Wochenschlüsseln kann nun der Kunde einem Energieberater Zugriff auf die Messdaten dieser Woche gewähren. D.h. der Kunde gewährt einem Energieberater den Zugriff Private-Keys der Wochenschlüssel für einen bestimmten Zeitraum, für welchen er die Messdaten weitergeben will. Dadurch kann der Energieberater nun die Messdatensätze des Kunden entschlüsseln und somit die Daten auswerten.

A.2 Abkürzungsverzeichnis

MD5 Message Digest 5

SHA Secure Hash Algorithm

DES Data Encryption Standard

AES Advanced Encryption Standard

IDEA International Data Encryption Algorithm

WCF Windows Communication Foundation

HTTP Hypertext Transfer Protocol

SOAP Simple Object Access Protocol

WSDL Web Service Description Language

IIS Internet Information Services

XML Extensible Markup Language

A.3 Listing der XML-Schemas des Datensätze eines Kunden

A.3.1 Kundeninformationsdatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="customer" nillable="true" type="customer" />
  <xs:complexType name="customer">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="salutation" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="firstname" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="lastname" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="email" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="streetNr" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="zipCity" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.3.2 Immobiliendatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="facility" nillable="true" type="facility" />
  <xs:complexType name="facility">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="name" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="gpsCoordinates" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="streetNr" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="zipCity" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="use" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="wayOfConstruction" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="buildingArea" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="usefulArea" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="floors" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="insulation" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="comment" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="keybandId" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.3.3 Dokumentendatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="document" nillable="true" type="document" />
  <xs:complexType name="document">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="name" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="filename" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="comment" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="documentData" type="xs:base64Binary" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.3.4 Zählerinformationsdatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="meter" nillable="true" type="meter" />
  <xs:complexType name="meter">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="name" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="type" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="automatic_manual" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="unit" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="meterId" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="comment" type="xs:string" />
    </xs:sequence>
```

```
  </xs:complexType>
</xs:schema>
```

A.3.5 Parameterdatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="meterParameter" nillable="true" type="meterParameter" />
  <xs:complexType name="meterParameter">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="username" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="name" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="type" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="unit" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="meterId" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="signatureKeyId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="signatureKey" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="keybandId" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="encryptionKeyId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="encryptionKey" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="encryptionKeyType" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="linkAddressDataStorage" type="xs:string" /
          >
      <xs:element minOccurs="0" maxOccurs="1" name="endpointDataStorage" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="linkAddressKeyManagement" type="xs:string"
          />
      <xs:element minOccurs="0" maxOccurs="1" name="endpointKeyManagement" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="linkAddressDataStorageClient" type="
          xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="meterMetadatasetId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="metadataPulbicKey" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="metadataPrivateKey" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="metadataKeyId" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.3.6 Messwertdatensatz

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="meteredValue" nillable="true" type="meteredValue" />
  <xs:complexType name="meteredValue">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="measuredValue" type="xs:float" />
      <xs:element minOccurs="0" maxOccurs="1" name="unit" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="timestamp" type="xs:dateTime" />
      <xs:element minOccurs="0" maxOccurs="1" name="status" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.4 Listing des XML-Schemas der Metadaten-Datenstruktur

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MetadataKeyInfo" nillable="true" type="MetadataKeyInfo" />
  <xs:complexType name="MetadataKeyInfo">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="KeyId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="KeyType" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="MetadataSetInfo" nillable="true" type="MetadataSetInfo" />
  <xs:complexType name="MetadataSetInfo">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Id" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="Tag" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="MetadataItem" nillable="true" type="MetadataItem" />
  <xs:complexType name="MetadataItem">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Type" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="Name" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="DataId" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="WeekInYear" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="Year" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="SignatureKeyId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="KeyInfoList" type="ArrayOfMetadataKeyInfo"
          />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfMetadataKeyInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="MetadataKeyInfo" nillable="true"
          type="MetadataKeyInfo" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Metadata" nillable="true" type="Metadata" />
  <xs:complexType name="Metadata">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Children" type="ArrayOfMetadata" />
      <xs:element minOccurs="0" maxOccurs="1" name="ChildrenMetadataSets" type="
          ArrayOfMetadataSetInfo" />
      <xs:element minOccurs="0" maxOccurs="1" name="Item" type="MetadataItem" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfMetadata">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Metadata" nillable="true" type="
          Metadata" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfMetadataSetInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="MetadataSetInfo" nillable="true"
          type="MetadataSetInfo" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.5 Beschreibung der Systemprozesse

Für allgemeine Informationen zu diesem Abschnitt sei auf Abschnitt 4.10 im Hauptteil der Arbeit verwiesen.

Dieser Abschnitt soll die verschiedenen Systemprozesse des Systems, vor allem aus der Sicht des Kunden und des Energieberaters zeigen. Aus diesem Grunde wird detailiert beschrieben, welche Aktionen der Kunde bzw. der Enerigeberater und dessen Clientapplikation ausführt.
Auf die detailiertere Beschreibung der Serverseite wird in den Abschnitten 4.11 bzw. A.6 eingegangen.

A.5.1 Ablauf der Systemprozesse

Der folgende Abschnitt beschreibt die Abläufe der einzelnen Systemprozesse. Dabei wird vor allem auf die auszuführenden Aktionen aus der Sicht des Kunden bzw. des Energieberaters eingegangen.

Die Systemprozesse werden in 6 Gruppen folgendermaßen aufgeteilt:

- Messwerterfassung durch eine automatische Datenquelle (Kapitel A.5.3)
- Dateneinbringung durch den Kunden (Kapitel A.5.4)
- Datenabfrage (Kapitel A.5.5)
- Datenanforderung und -freigabe (Kapitel A.5.6)
- Editieren von Datenobjekten (Kapitel A.5.7)
- Sonstige Systemprozesse (Kapitel A.5.8)

A.5.2 Erklärung der verwendeten Symbole

Abbildung A.1 stellt die in den anschließend folgenden Ablaufbeschreibungen verwendeten Symbole dar und erklärt deren Bedeutung.

A.5.3 Messwerterfassung durch eine automatische Datenquelle

Systemprozess 1.1 Automatische Messdatenerfassung (online)

Dieser Systemprozess ist im Hauptteil der Arbeit in Abschnitt 4.10.2 beschrieben.

Systemprozess 1.2 Automatische Messdatenerfassung (offline)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung

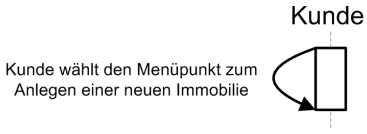
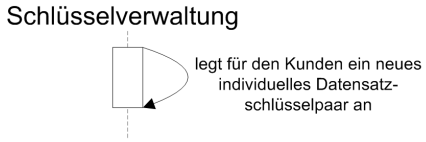
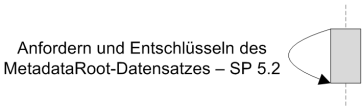
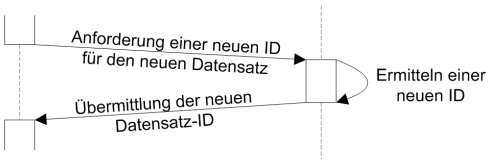
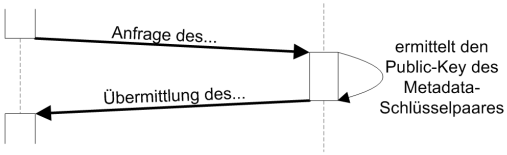
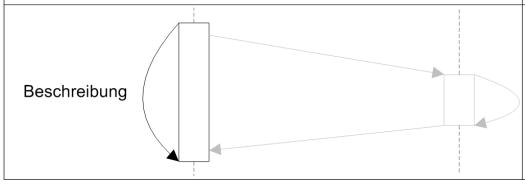
Beispiel	Erklärung
	Dieses Symbol zeigt eine Aktion welche vom Kunden oder Berater (hier Kunde) selbst ausgeführt wird.
	Dieses Symbol zeigt eine Aktion welche von der entsprechenden Partei (hier Schlüssel-verwaltung) ausgeführt wird. Sollte es sich dabei um den Kunden oder den Berater handeln, so wird die Aktion vom Kundenclient bzw. Beraterclient ausgeführt.
	Dieses Symbol zeigt die Ausführung eines anderen Systemprozesses (hier Systemprozess 5.2).
	Dieses Symbol zeigt einen einfachen Request, welcher eine Aktion auf der Serverseite aufruft. Schlussendlich wird die Aktion mit einem Response beantwortet.
	Dieses Symbol zeigt eine mehrfache Request-Response Aktion. Dies bedeutet eine serielle Ausführung mehrerer einfacher Requests, von welchen jeder eine Aktion auf der Serverseite auslöst und mit einem Response beantwortet wird.
	Dieses Symbol fasst mehrere Aktionen zusammen, wobei dabei auch mehrere Requests an die entsprechende Serverseite gesendet werden können.

Abbildung A.1: Erklärung der verwendeten Symbole in den Systemprozesse

Systemprozess: 1.2 Automatische Messdatenerfassung (offline)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Datenquelle ist offline (die Schlüsselverwaltung und der Datenspeicher sind **nicht** erreichbar); die Datenquelle hat einen neuen Messwert und den zugehörigen Messzeitpunkt; die Datenquelle ist richtig parametriert
Kurzbeschreibung: Die Datenquelle buffert die Messdaten. Solange der Wochenschlüssel (Public Key) noch aktuell ist, werden die Daten verschlüsselt in der Datenquelle gespeichert. Ist kein aktueller Schlüssel mehr vorhanden, werden die Messwertdatensätze im Klartext in der Datenquelle gespeichert.

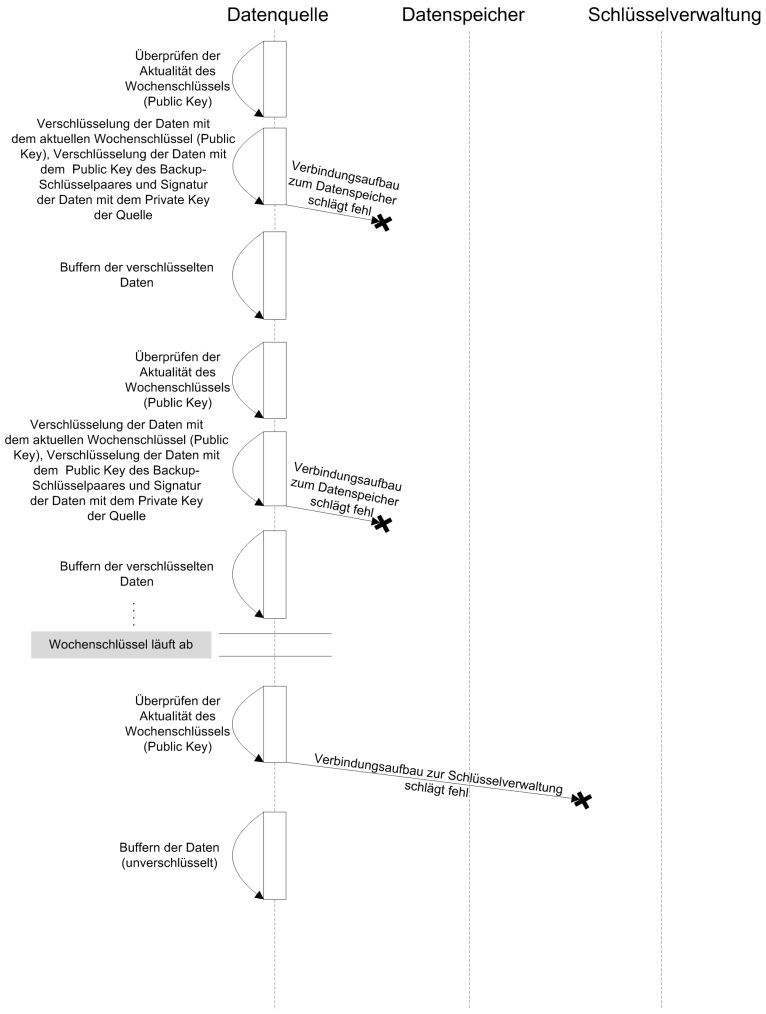


Abbildung A.2: Systemprozess 1.2 Automatische Messdatenerfassung (offline)

Voraussetzungen:

- die Datenquelle ist offline (die Schlüsselverwaltung und der Datenspeicher sind **nicht** erreichbar)
- die Datenquelle hat einen neuen Messwert und den zugehörigen Messzeitpunkt
- die Datenquelle ist richtig parametriert

Beschreibung: Nachdem die automatische Datenquelle (z.B. ein automatischer Stromzähler) den Messwert ermittelt hat, wird zuerst überprüft, ob der aktuelle Wochenschlüssel schon im Zähler vorhanden ist. Da hier davon ausgegangen wird, dass die Datenquelle den aktuellen Wochenschlüssel schon kennt, verschlüsselt die Datenquelle den neuen Messwertdatensatz mit einem neuen Session-Key. Dieser wird anschließend mit dem Public-Key des aktuellen Wochenschlüssels und mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Danach versucht die Datenquelle eine Verbindung zum Datenspeicher aufzubauen, um eine neue Datensatz-ID anzufordern. Da die Datenquelle über keine Internetverbdinung verfügt, schlägt der Verbindungsaufbau fehl. Dies hat zur Folge, dass der verschlüsselte Messwertdatensatz bis zur Wiedererlangung einer Verbindung in der Datenquelle gebuffert wird. Der bisher beschriebene Vorgang wird solange wiederholt, bis der aktuelle Wochenschlüssel abläuft (oder die Internetverbindung wiederhergestellt wird (siehe hierzu Systemprozess 1.3 - Abschnitt A.5.3).

Angenommen die Internetverbindung kann bis zum Ablauf des aktuellen Wochenschlüssels nicht wiederhergestellt werden. In diesem Fall versucht die automatische Datenquelle die Verbindung zur Schlüsselverwaltung herzustellen, um einen neuen Wochenschlüssel anzufordern. Da dies fehlschlägt, werden alle nachfolgenden Messwerte der Datenquelle unverschlüsselt gebuffert.

Systemprozess 1.3 Automatische Messdatenerfassung (offline → online)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar)
- die Datenquelle hat einige gebufferte Datensätze, welche aufgrund veralteter Schlüssel nur teilweise unverschlüsselt sind
- die Datenquelle ist richtig parametriert

Beschreibung: Sobald die Datenquelle die Internetverbindgung wiedererlangt werden alle Wochenschlüssel der noch nicht verschlüsselten Datensätze bei der Schlüsselverwaltung angefordert (dieser Vorgang erfordert einen Request pro Schlüssel). Dabei werden die Requests mit dem Private-Key des Signaturschlüssels, welcher in der Datenquelle parametriert ist, signiert. Die Schlüsselverwaltung prüft die Signatur mithilfe des entsprechenden Public-Keys und liefert anschließend die geforderten Wochenschlüssel. Danach erzeugt die Datenquelle für jeden Messwertdatensatz einen

Systemprozess: 1.3 Automatische Messdatenerfassung (Wiedererlangung der Internetverbindung der Datenquelle – „offline → online“)

beteiligte Parteien: Datenquelle, Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar) und hat einige gebufferte Datensätze, welche aufgrund veralteter Schlüssel nur teilweise unverschlüsselt sind; die Datenquelle ist richtig parametriert
Kurzbeschreibung: Die Datenquelle fordert bei der Schlüsselverwaltung alle zur Verschlüsselung der gebufferten Daten notwendigen Wochenschlüssel an (inklusive dem aktuellen Wochenschlüssel). Danach werden alle gebufferten Daten, welche noch nicht verschlüsselt sind, mit dem entsprechenden Wochenschlüssel und mit dem Public Key des Backup-Schlüsselpaares verschlüsselt und beim Datenspeicher neue Datensatz-IDs angefordert. Nachdem die Datenquelle das Kunden-Lock angefordert hat, liest diese den Metadatensatz des Zählers und legt neue Metadatensätze an bzw. aktualisiert diese (trägt die neuen Messwerte ein). Danach werden die Daten im Datenspeicher gesichert und das Lock gelöst.

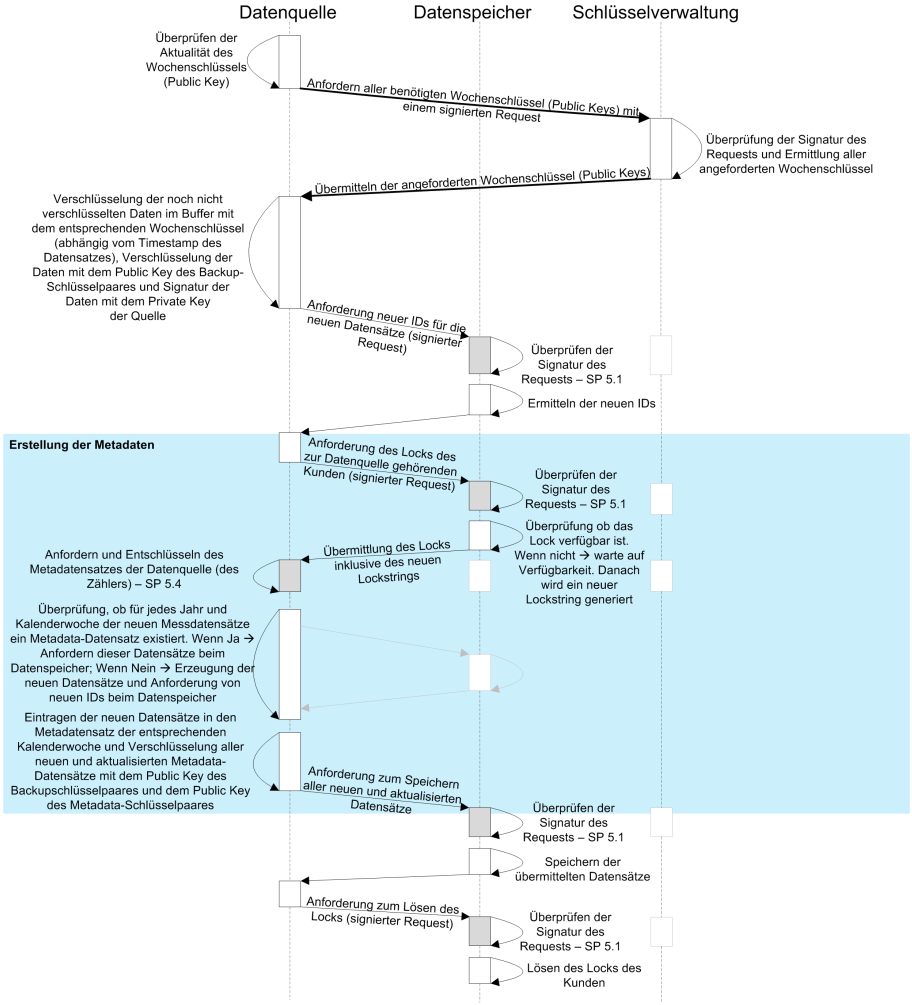


Abbildung A.3: Systemprozess 1.3 Automatische Messdatenerfassung (offline → online)

neuen Session-Key und verschlüsselt diesen mit dem entsprechenden Public-Key des Wochenschlüssels (abhängig vom Messzeitpunkt) und dem Public-Key des Backup-Schlüsselpaares, welcher in der Datenquelle parametriert ist. Anschließend werden neue Datensatz-IDs für die neuen Messwertdatensätze beim Datenspeicher angefordert. Nach der Überprüfung der Signatur des Requests (siehe Systemprozess 5.1 - Abschnitt A.5.8) liefert der Datenspeicher die geforderten neuen Datensatz-IDs.

Danach fordert die Datenquelle das Lock des Kunden beim entsprechenden Datenspeicher an. Nachdem dieser die Signatur des Requests überprüft hat, wird bei Erhalt des Locks der Lockstring an die Datenquelle übermittelt. Anschließend fordert die Datenquelle den Metadata-Datensatz des Zählers beim Datenspeicher an (siehe Systemprozess 5.4 - Abschnitt A.5.8). Anschließend überprüft die Datenquelle, ob für jeden Messwertdatensatz ein Metadata-Datensatz des entsprechenden Jahres und der entsprechenden Kalenderwoche schon existieren. Sollte dies der Fall sein, so werden diese beim Datenspeicher angefordert (siehe Abschnitt A.5.8). Andernfalls werden diese angelegt und dazu auch neue Datensatz-IDs beim Datenspeicher angefordert (wie oben schon für die Messwertdatensätze). Anschließend werden alle gebufferten Messwertdatensätze in den Metadata-Datensatz der entsprechenden Kalenderwoche eingetragen. Anschließend werden alle neuen oder geänderten Metadata-Datensätze mit einem neuen Session-Key symmetrisch verschlüsselt und dieser Session-Key mithilfe des Public-Keys des Metadata-Schlüsselpaares (in der Datenquelle parametriert) und des Public-Keys des Backup-Schlüsselpaares verschlüsselt. Anschließend werden alle neuen oder aktualisierten Datensätze an den Datenspeicher geschickt. Dieser speichert die Daten in der Datenbank, nachdem die Signatur des Requests überprüft wurde. Anschließend fordert die Datenquelle die Lösung des Kundenlocks beim Datenspeicher an. Dieser überprüft wiederum die Signatur und löst anschließend das Lock des Kunden.

A.5.4 Dateneinbringung durch den Kunden

Systemprozess 1.4 Anlegen einer Immobilie durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)

Beschreibung: Nachdem der Kunde den Menüpunkt zum Anlegen einer neuen Immobilie gewählt hat, gibt er die deskriptiven Daten zur Immobilie (Art der Nutzung, Art der Heizung, Grundfäche, Adresse,...) ein. Anschließend wird ein neues Schlüsselband für die neue Immobilie bei der Schlüsselverwaltung angefordert. Nachdem die Schlüsselverwaltung ein neues Schlüsselband für den Kunden angelegt hat, wird die Schlüsselband-ID an den Kunden zurückgeliefert. Diese Schlüsselband-ID wird im neuen Immobiliendatensatz eingetragen und anschließend wird ein neuer individueller Datensatzschlüssel für den Immobiliendatensatz bei der Schlüsselverwaltung angefordert. Die Schlüsselverwaltung erstellt ein neues Schlüsselpaar und liefert den Public-Key dieses neuen Schlüsselpaares zurück. Anschließend wird noch der Public-Key

Systemprozess: 1.4 Anlegen einer Immobilie durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
Kurzbeschreibung: Kunde wählt den Menüpunkt zum Anlegen einer neuen Immobilie und gibt die entsprechenden Daten für diese ein (Art der Immobilie, Nutzung, Fläche, Art der Heizung, Ort,...). Anschließend wird bei der Schlüsselverwaltung ein neues Schlüsselband, ein neuer individueller Datensatzschlüssel und der Public-Key des Backup-Schlüsselpaares angefordert. Danach wird der neue Datensatz mit diesen beiden Schlüsseln verschlüsselt. Nachdem eine neue ID für den Datensatz ermittelt und das Lock des Kunden beim Datenspeicher angefordert wurde, wird der neue Datensatz im MetadataRoot-Datensatz hinzugefügt. Danach wird der MetadataRoot-Datensatz wieder mit dem Public-Key des Backup-Schlüsselpaares und dem Public-Key des Metadata-Schlüsselpaares verschlüsselt (welcher auch angefordert wurde). Danach werden die Daten im Datenspeicher gesichert und das Lock wieder gelöst.

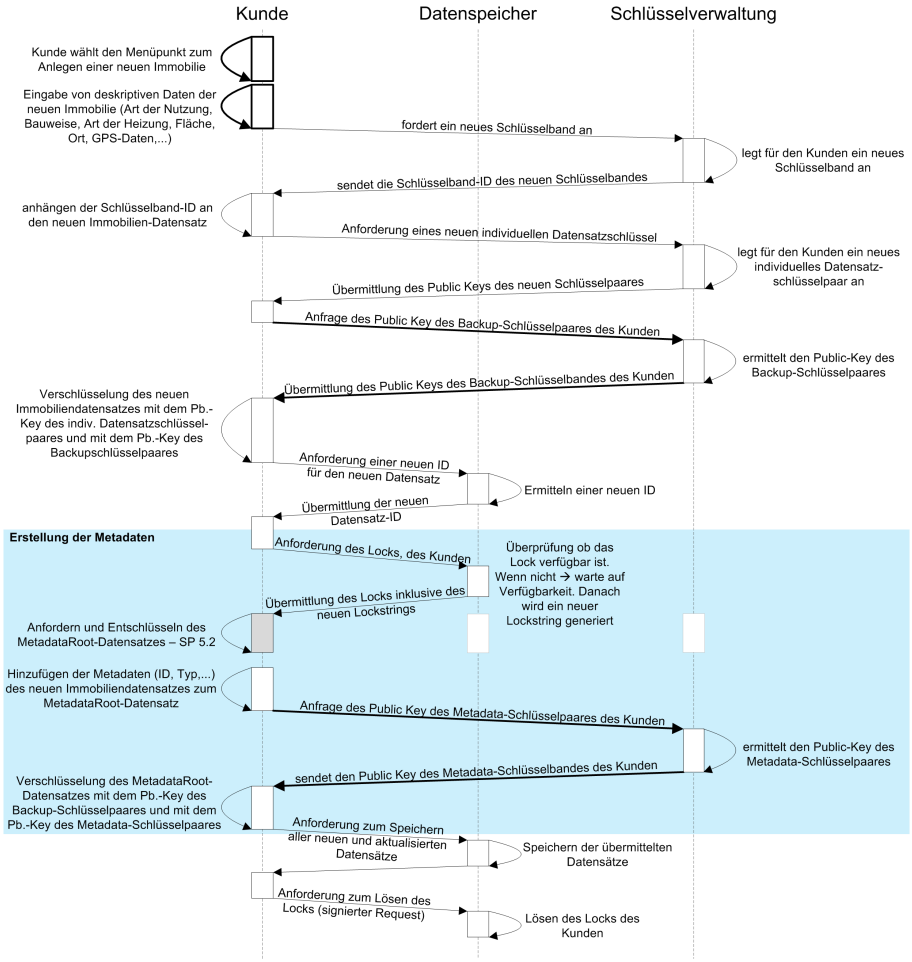


Abbildung A.4: Systemprozess 1.4 Anlegen einer Immobilie durch den Kunden

des Backup-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Hierzu wird zuerst eine Liste aller Schlüssel des Typs „backup“ angefordert um die Schlüssel-ID herauszufinden und danach wird der Schlüssel selbst angefordert. Sobald dieser zurückgeliefert wurde, wird ein neuer Session-Key erzeugt, mit welchem der Immobliliendatensatz symmetrisch verschlüsselt wird. Anschließend wird dieser Session-Key mithilfe des Public-Keys des neuen individuellen Schlüsselpaares und mithilfe des Public-Keys des Backup-Schlüsselpaares asymmetrisch verschlüsselt. Danach wird beim Datenspeicher eine neue Datensatz-ID für den neuen Immobiliendatensatz angefordert.

Anschließend wird das Lock des Kunden beim Datenspeicher angefordert. Dieser sendet, sobald das Lock verfügbar ist, den Lockstring zurück. Danach wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8), in welchen der neue Immobiliendatensatz eingetragen wird. Hiernach wird der Public-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Diese Anforderung besteht aus der Anforderung einer Liste aller Schlüssel-IDs des Kunden mit dem Typ „metadata“ und der Anforderung des Schlüssels selbst. Nach dem Erhalt des Schlüssels wird der MetadataRoot-Datensatz mit einem neu generierten Session-Key verschlüsselt. Dieser Session-Key wird dann mit dem Public-Key des Metadata-Schlüsselpaares und dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Danach werden der neue Immobiliendatensatz und der aktualisierte MetadataRoot-Datensatz im Datenspeicher gespeichert. Abschließend wird das Lock des Kunden im Datenspeicher gelöst.

Systemprozess 1.5 Inventarisierung eines Zählers

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Kunde hat zumindest eine Immobilie angelegt

Beschreibung: Nachdem der Kunde den Menüpunkt zum Inventarisieren eines neuen Zählers gewählt hat, wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe dazu Systemprozess 5.2 - Abschnitt A.5.8). Dieser wird benötigt, um dem Kunden eine Liste seiner Immobilien anzuzeigen. Anschließend wählt der Kunde eine Immobilie aus, für welche ein neuer Zähler inventarisiert werden soll. Danach wählt der Kunde, ob es sich bei dem neuen Zähler um einen automatischen oder manuellen Zähler handeln soll. Anschließend gibt der Kunde deskriptive Daten zum Zähler (Messwerteinheit, Name des Zählers, Art des Zählers,...) ein. Hierauf wird der Public-Key des Backupschlüsselpaares bei der Schlüsselverwaltung angefordert. Hierzu wird zuerst eine Liste aller Schlüssel des Typs „backup“ angefordert, um die Schlüssel-ID herauszufinden und danach wird der Schlüssel selbst angefordert. Sobald der Schlüssel übermittelt wurde, wird der Zählerdatensatz mit einem neuen Session-Key symmetrisch verschlüsselt und anschließend der Session-Key selbst mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt.

Systemprozess: 1.5 Inventarisierung eines Zählers (Teil 1)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); der Kunde hat zumindest eine Immobilie angelegt
Kurzbeschreibung: Kunde wählt den Menüpunkt zum Anlegen eines neuen Zählers. Nachdem der MetadataRoot-Datensatz geladen und entschlüsselt wurde, wählt der Kunde eine Immobilie aus. Danach wählt der Kunde, ob es sich um einen automatischen oder manuellen Zähler handelt und gibt deskriptive Daten ein. Danach wird der neue Zählerdatensatz mit dem angeforderten Public-Key des Backup-Schlüsselpaares verschlüsselt. Dann wird der entsprechende Immobiliendatensatz geholt, entschlüsselt und die Schlüsselband-ID gefiltert. Danach wird der Zähler mit dem Public-Key von zumindest einem Schlüssel dieses Schlüsselbandes auf welche andere Benutzer Zugriff haben verschlüsselt. Fortsetzung in Teil 2.

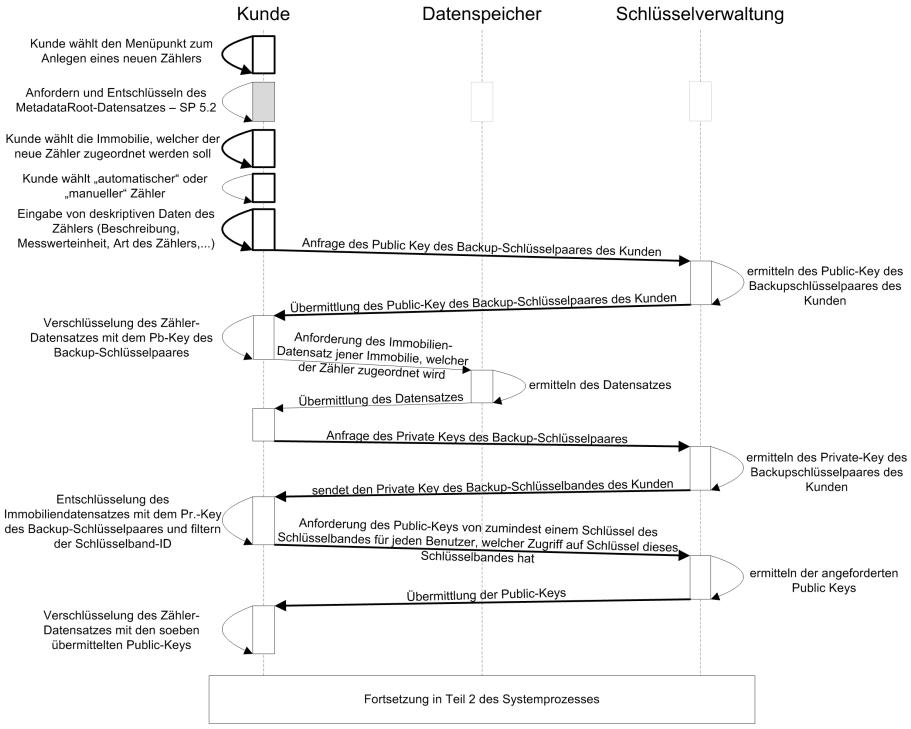


Abbildung A.5: Systemprozess 1.5 Inventarisierung eines Zählers (Teil 1)

Systemprozess: 1.5 Inventarisierung eines Zählers (Teil 2)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar), Teil 1 des Use-Cases wurde ausgeführt
Kurzbeschreibung: Nach der Ausführung von Teil 1 werden neue Datensatz-IDs beim Datenspeicher angefordert. Danach wird das Lock des Kunden geholt, der MetadataRoot-Datensatz gelesen und ein neuer Metadatensatz für den Zähler angelegt. Wenn es sich beim neuen Zähler um einen automatischen Zähler handelt, werden als nächstes alle für den Parameterdatensatz notwendigen Schlüssel angefordert und anschließend ein neuer Parameterdatensatz erstellt und die zugehörigen Metadaten erstellt. Danach werden alle neuen und aktualisierten Metadatensätze verschlüsselt und alle Daten im Datenspeicher gesichert. Abschließend wird das Lock des Kunden gelöst.

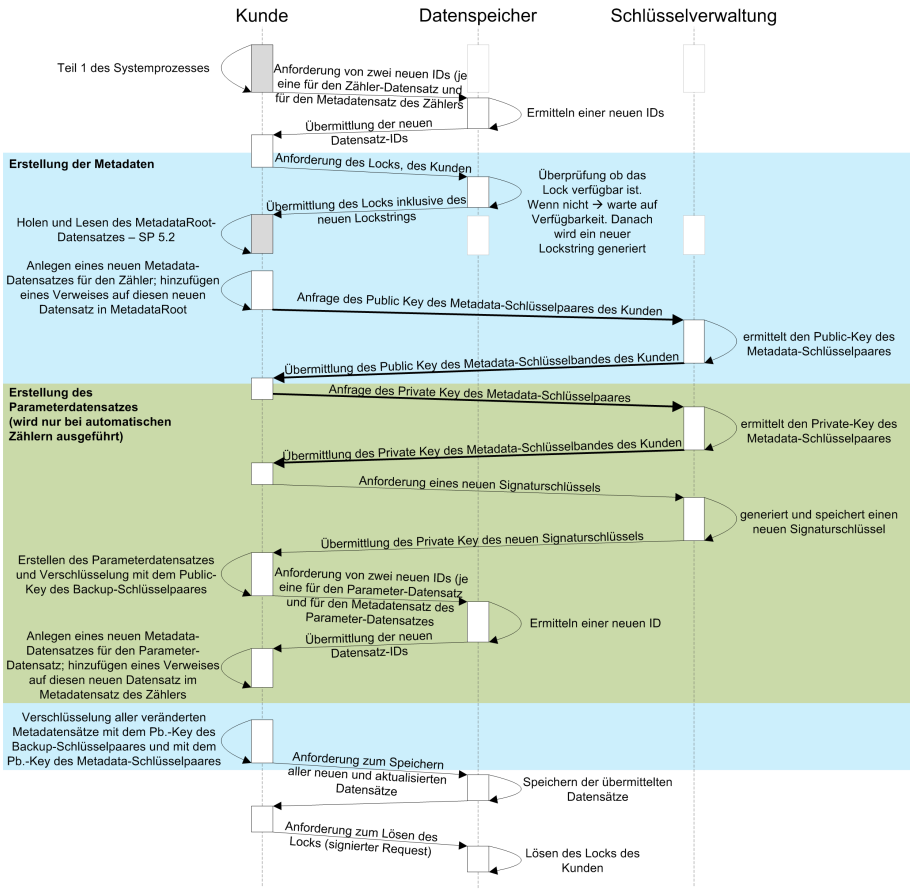


Abbildung A.6: Systemprozess 1.5 Inventarisierung eines Zählers (Teil 2)

Anschließend wird der Immobilien-Datensatz jener Immobilie, welcher der neue Zähler zugeordnet wird, beim Datenspeicher angefordert. Danach wird der Private-Key des Backup-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Diese Anforderung besteht wiederum aus der Anforderung einer Schlüsselliste und der anschließenden Anforderung des Schlüssels selbst. Der Schlüssel wird dann verwendet um den zuvor angeforderten Immobiliendatensatz zu entschlüsseln. Aus diesem Immobiliendatensatz kann nun die Schlüsselband-ID der Immobilie herausgefunden werden.

Danach wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, welche für jeden Berater, dem Schlüssel dieses Schlüsselbandes freigegeben wurden, zumindest die Schlüssel-ID eines dieser Schlüssel enthält. Danach werden die Public-Keys jedes einzelnen dieser Schlüsselpaare angefordert. Anschließend wird der neue Zählerdatensatz mit einem neuen Session-Key verschlüsselt. Daraufhin wird der Session-Key mithilfe des Public-Keys des Backup-Schlüsselpaares und mit allen Public-Keys der gerade angeforderten Wochenschlüssel verschlüsselt. Dies dient dazu, dass jeder Benutzer, welcher Zugriff auf Schlüssel dieses Schlüsselbandes hat, auch den neuen Zählerdatensatz einsehen kann.

Nachdem der neue Zählerdatensatz verschlüsselt wurde, werden zwei neue Datensatz-IDs beim Datenspeicher angefordert. Hierbei dient je eine ID für den neuen Zählerdatensatz und eine ID für den neuen Metadata-Datensatz welcher die Metadaten für den neuen Zähler enthält. Danach wird das Lock des Kunden beim Datenspeicher angefordert. Sobald das Lock verfügbar ist, wird der Lockstring übermittelt. Anschließend wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Im Anschluss wird ein neuer Metadata-Datensatz für die Metadaten des neuen Zählers erzeugt und ein Verweis auf diesen Metadata-Datensatz im MetadataRoot-Datensatz eingetragen. Danach wird der Public-Key des Metadata-Schlüssels bei der Schlüsselverwaltung angefordert. Diese Anforderung besteht aus der Anforderung einer Liste aller Schlüssel-IDs des Kunden des Typs „metadata“ und der Anforderung des Schlüssels selbst.

Als nächstes wird der Parameter-Datensatz erzeugt. Dieser Schritt wird **nur bei automatischen Zählern** ausgeführt. Für den Parameter-Datensatz wird als erstes der Private-Key des Metadata-Schlüsselpaares angefordert. Diese Anforderung besteht wie zuvor aus der Anforderung einer Liste von Schlüssel-IDs und der Anforderung des Schlüssels selbst. Danach wird bei der Schlüsselverwaltung ein neuer Signaturschlüssel für den Zähler angefordert. Hierbei erzeugt die Schlüsselverwaltung ein neues Schlüsselpaar und retourniert dessen Private-Key. Nachdem nun alle, für die Erstellung des Parameter-Datensatzes notwendigen Daten vorhanden sind, kann dieser nun erstellt werden. Anschließend wird dieser mit dem am Anfang angeforderten Public-Key des Backup-Schlüsselpaares verschlüsselt. Anschließend werden wieder zwei neue Datensatz-IDs beim Datenspeicher angefordert. Hierbei dient wiederum eine für den Parameter-Datensatz und eine für den Metadata-Datensatz des Parameterdatensatzes, welcher im Anschluss erzeugt wird. Danach wird ein Verweis auf den Metadata-Datensatz des Parameterdatensatzes im Metadata-Datensatz des Zählers eingetragen.

Anschließend werden alle geänderten und neuen Metadata-Datensätze mit dem Public-Key des Metadata-Schlüsselpaares und mit dem Public-Key des Backup-Schlüs-

selpaares verschlüsselt. Danach werden alle neuen und aktualisierten Datensätze an den Datenspeicher geschickt, um diese dort zu speichern. Abschließend wird noch das Lock des Kunden gelöst.

Systemprozess 1.6 Händische Messdatenerfassung

Systemprozess: 1.6 Händische Messdatenerfassung (Teil 1)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar), der Kunde hat zumindest eine Immoblilie angelegt, zu welcher ein Zähler inventarisiert wurde
Kurzbeschreibung: Nachdem der Kunde den Menüpunkt zur Messdatenerfassung gewählt hat, wird der MetadataRoot-Datensatz gelesen. Anschließend wählt der Kunde die entsprechende Immobilie und den Zähler. Danach wird der Metadatensatz des Zählers geladen und anschließend der Zählerdatensatz selbst. Danach wird zuerst der Public-Key des Backupschlüsselpaares angefordert und dann der Immobiliendatensatz geholt und entschlüsselt um die Schlüsselband-ID zu ermitteln. Anschließend wird der entsprechende Wochenschlüssel angefordert. Anschließend wird der neue Messwertdatensatz mit dem Public-Key des Backup-Schlüsselpaares und mit dem Public-Key des Wochenschlüssels verschlüsselt. Fortsetzung in Teil 2.

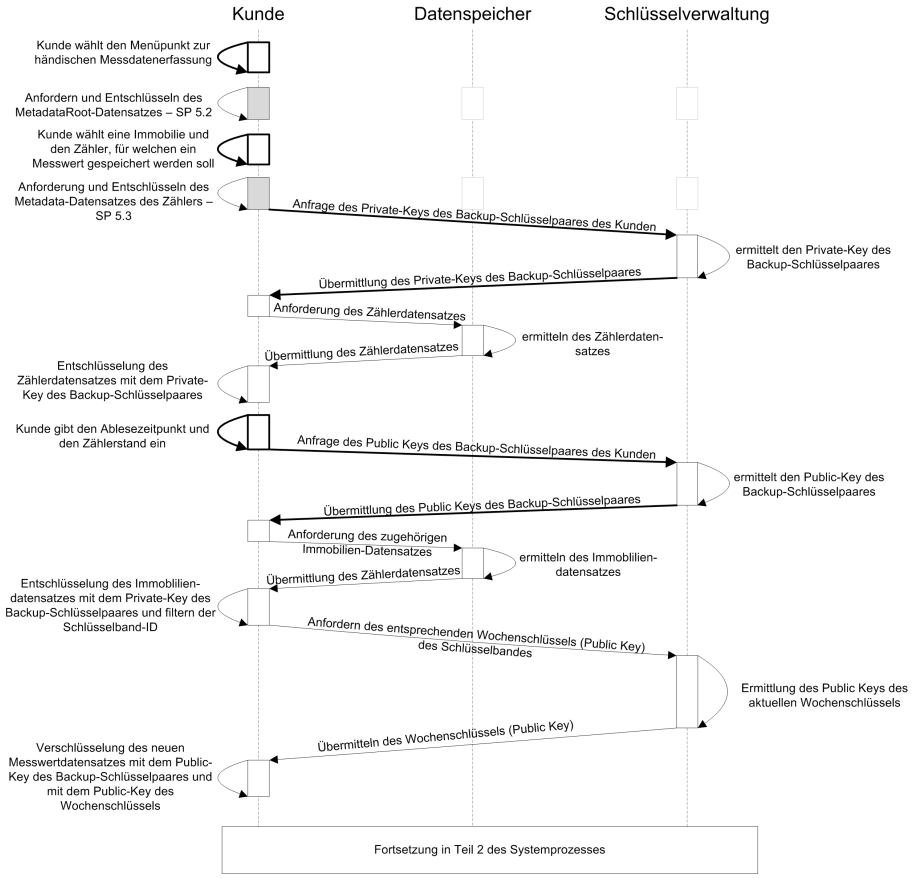


Abbildung A.7: Systemprozess 1.6 Inventarisierung eines Zählers (Teil 1)

Systemprozess: 1.6 Händische Messdatenerfassung (Teil 2)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar), der Kunde hat zumindest eine Immoblilie angelegt, zu welcher ein Zähler inventarisiert wurde, Teil 1 wurde ausgeführt
Kurzbeschreibung: Nach der Ausführung von Teil 1 wird zuerst eine neue Datensatz-ID und das Lock des Kunden beim Datenspeicher angefordert. Danach wird der Metadatensatz des Zählers gelesen. Anschließend wird ermittelt, ob es für die entsprechende Kalenderwoche schon einen Metadatensatz gibt und dieser bei Bedarf angelegt. Danach wird der neue Messwertdatensatz im Metadatensatz der Kalenderwoche vermerkt. Anschließend wird der Public-Key des Meta-Datensatzes angefordert und alle geänderten oder neuen Metadatensätze mit diesem und dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Abschließend werden alle neuen und aktualisierten Datensätze im Datenspeicher gesichert und das Lock gelöst.

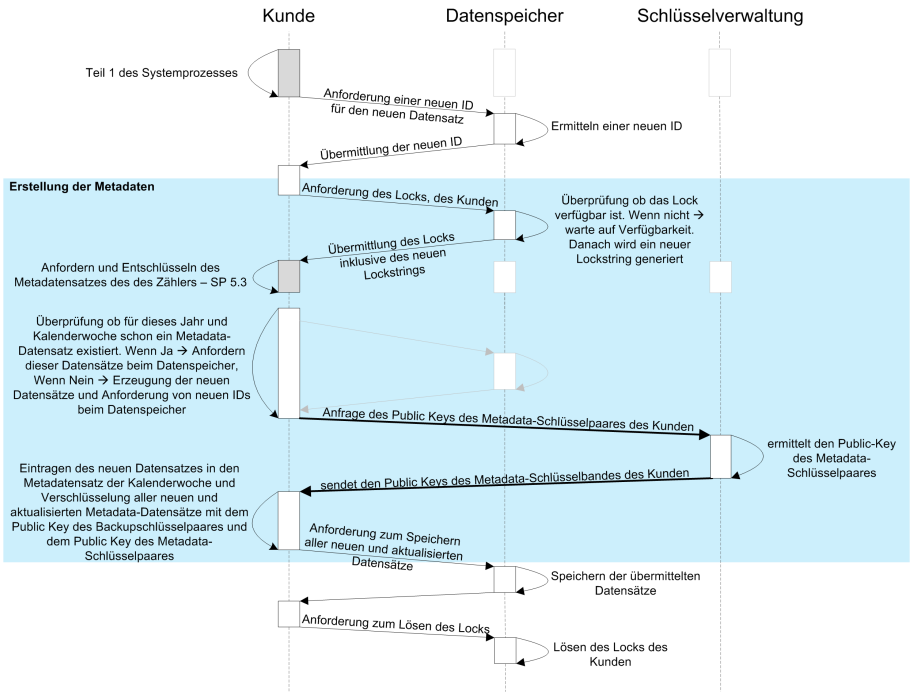


Abbildung A.8: Systemprozess 1.6 Inventarisierung eines Zählers (Teil 2)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Kunde hat zumindest eine Immobilie angelegt, zu welcher ein Zähler inventarisiert wurde

Beschreibung: Nachdem der Kunde den Menüpunkt zur händischen Messdatenerfassung gewählt hat, wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe dazu Systemprozess 5.2 - Abschnitt A.5.8). Daraus erhält der Kunde eine Liste seiner gespeicherten Immobilien und Zähler. Aus dieser Liste wählt der Kunde die gewünschte Immobilie und den gewünschten Zähler aus. Anschließend wird der Metadata-Datensatz des Zählers angefordert und entschlüsselt (siehe dazu Systemprozess 5.3 - Abschnitt A.5.8). Der Grund, warum der Kunde schon einen Zähler wählen kann, obwohl der Metadata-Datensatz des Zählers zu diesem Zeitpunkt noch gar nicht geladen ist, ist jener, dass der Name des Zählers als *Tag* im Verweis im MetadataRoot-Datensatz verwendet wird.

Als nächstes wird bei der Schlüsselverwaltung der Private Key des Backup-Schlüsselpaares angefordert. Diese Anfrage besteht aus einer Anfrage nach einer Liste von Schlüssel-IDs des Kunden vom Typ „backup“ und der Anfrage nach dem Schlüssel selbst. Anschließend wird der Zählerdatensatz beim Datenspeicher angefordert. Nachdem der Datensatz geliefert wurde, wird dieser mit dem Private-Key des Backup-Schlüsselpaares entschlüsselt. Nun kann dem Kunden die erwartete Messwerteinheit und weitere Informationen zum Zähler angezeigt werden. Als nächstes gibt der Kunde den Zählerstand und den Ablesezeitpunkt ein. Danach wird der Public-Key des Backup-Schlüsselpaares angefordert. Wie zuvor besteht diese Anforderung wieder aus einer Auflistung von Schlüssel-IDs und der Anforderung des Schlüssels selbst.

Als nächstes muss der aktuelle Wochenschlüssel des zugehörigen Schlüsselbandes angefordert werden. Dazu wird zuerst beim Datenspeicher der entsprechende Immobiliendatensatz angefordert und anschließend mit dem Private-Key des Backup-Schlüsselpaares entschlüsselt. Anschließend kann aus dem Datensatz die Schlüsselband-ID herausgelesen werden. Danach wird der zum Ablesezeitpunkt gehörende Public-Key des Wochenschlüssels bei der Schlüsselverwaltung angefordert. Nun wird der neu erzeugte Messwertdatensatz mit einem neuen Session-Key verschlüsselt. Dieser Session-Key wird anschließend sowohl mit dem Public-Key des Backup-Schlüsselpaares, als auch mit dem Public-Key des Wochenschlüssels verschlüsselt.

Als nächstes wird eine neue Datensatz-ID für den neuen Messwertdatensatz beim Datenspeicher angefordert. Anschließend wird das Lock des Kunden angefordert. Sobald dieses verfügbar ist, sendet der Datenspeicher den Lockstring. Anschließend wird der Metadata-Datensatz des Zählers angefordert und entschlüsselt (siehe Systemprozess 5.3 - Abschnitt A.5.8). Danach wird überprüft ob für das zum Ablesezeitpunk gehörende Jahr und Kalenderwoche schon Metadata-Datensätze existieren. Sollte dies der Fall sein, so werden diese beim Datenspeicher angefordert (siehe Systemprozess 5.3 Abschnitt A.5.8). Andernfalls werden diese angelegt und dazu auch neue Datensatz-IDs beim Datenspeicher angefordert (wie oben schon für den

Messwertdatensatz). Anschließend wird der neue Messwertdatensatz in den Metadata-Datensatz der entsprechenden Kalenderwoche eingetragen. Danach wird der Public-Key des Metadata-Schlüsselpaares angefordert (wiederum zwei Requests - einer zur Auflistung der Schlüssel-IDs und einer für den Schlüssel). Anschließend werden alle neuen oder geänderten Metadata-Datensätze mit einem neuen Session-Key symmetrisch verschlüsselt und der Session-Key mithilfe des Public-Keys des Metadata-Schlüsselpaares und des Public-Keys des Backup-Schlüsselpaares verschlüsselt. Anschließend werden alle neuen oder aktualisierten Datensätze an den Datenspeicher geschickt, welcher die Datensätze speichert. Abschließend wird das Lock des Kunden im Datenspeicher gelöst.

Systemprozess 1.7 Anlegen eines Kundeninformationsdatensatzes

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- es existiert noch kein Kundeninformationsdatensatz (in keinem Datenspeicher) (Überprüfung durch Systemprozess 5.5 - Abschnitt A.5.8)

Beschreibung: Nachdem der Kunde den Menüpunkt zum Anlegen eines neuen Kundeninformationdsdatensatzes gewählt hat wird überprüft, ob in irgendeinem Datenspeicher schon ein Kundeninformationsdatensatz zu diesem Kunden gespeichert ist. Diese Überprüfung geschieht durch Ausführung von Systemprozess 5.5 (Abschnitt A.5.8). Sollte schon ein solcher Datensatz vorhanden sein, wird der komplette Vorgang abgebrochen und der Kunden kann keinen neuen Kundeninformationsdatensatz anlegen. Andernfalls kann der Kunde seine Daten (Vorname, Nachname, Adresse, Email,...) eingeben. Anschließend wird bei der Schlüsselverwaltung ein neuer individueller Datensatzschlüssel angefordert. Nachdem die Schlüsselverwaltung den neuen Schlüssl angelegt hat, wird der Public-Key dieses Schlüsselpaares zurückgeschickt. Danach wird der Public-Key des Backup-Schlüsselpaares angefordert. Hierzu wird zuerst eine Liste von Schlüssel-IDs des Kunden mit dem Typ „backup“ und anschließend der Schlüssel selbst angefordert. Nach dem Erhalt des Schlüssels wird ein neuer Session-Key erzeugt und der Kundeninformationsdatenatz mit diesem Session-Key verschlüsselt. Anschließend wird der Session-Key mit dem Public-Key des Backup-Schlüsselpaares und mit dem Public-Key des neu generierten individuellen Schüsselpaares verschlüsselt. Danach wird eine neue Datensatz-ID beim Datenspeicher angefordert.

Als nächstes wird das Lock des Kunden beim Datenspeicher angefordert. Sobald dieses verfügbar ist, wird der Lockstring zurückgeliefert. Hiernach wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8) und der neue Kundeninformationsdatensatz eingetragen. Nun wird der Public-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert (zwei Requests: einer für die Auflistung der Schlüssel-IDs und einer für den

Systemprozess: 1.7 Anlegen eines Kundeninformationsdatensatzes

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); es existiert noch kein Kundeninformationsdatensatz (in keinem Datenspeicher) (Überprüfung durch Systemprozess 5.5)
Kurzbeschreibung: Der Kunde wählt den Menüpunkt Anlegen eines Kundeninformationsdatensatzes. Danach wird überprüft, ob in keinem Datenspeicher zu diesem Kunden ein Kundeninformationsdatensatz gespeichert wurde. Sollte dies der Fall sein, so wird der Vorgang abgebrochen. Andernfalls kann der Kunde seine Informationen eingeben. Anschließend werden ein neuer Datensatzschlüssel und der Public Key des Backup-Schlüsselpaares angefordert, mit welchen der neue Datensatz verschlüsselt wird. Anschließend wird eine neue Datensatz-ID und das Lock des Kunden beim Datenspeicher angefordert. Danach wird der MetadataRoot-Datensatz gelesen und der neue Datensatz eingetragen. Anschließend wird dieser wieder mit dem Public Key des Metadata-Schlüsselpaares und mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Zum Schluss werden alle neuen und aktualisierten Datensätze im Datenspeicher gesichert und das Lock des Kunden gelöst.

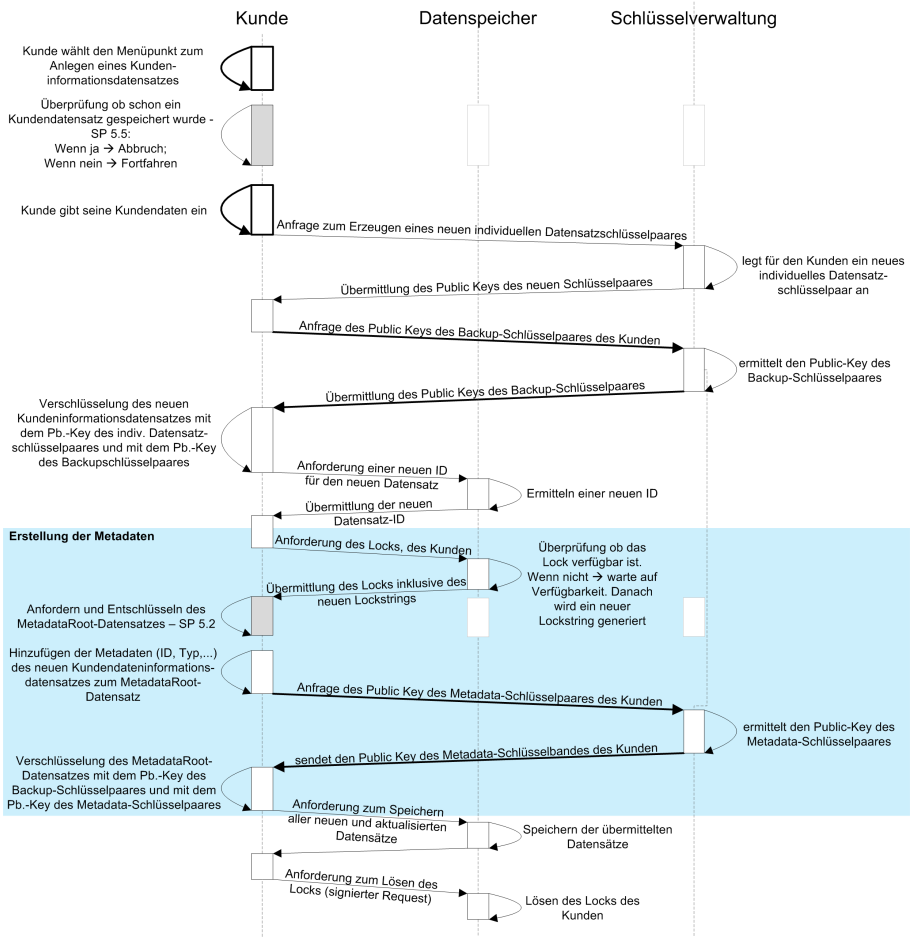


Abbildung A.9: Systemprozess 1.7 Anlegen eines Kundeninformationsdatensatzes

Schlüssel selbst). Danach wird der MetadataRoot-Datensatz mit einem neuen Session-Key verschlüsselt und dieser Session-Key mit dem Public-Key des Backup-Schlüsselpaares und mit dem Public-Key des Metadata-Schlüsselpaares verschlüsselt. Anschließend werden der aktualisierte MetadataRoot-Datensatz und der neue Kundeninformationsdatensatz im Datenspeicher gespeichert und das Lock des Kunden gelöst.

Systemprozess 1.8 Upload eines Dokuments durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)

Beschreibung: Als erstes wählt der Kunde den Menüpunkt zum Upload eines neuen Dokuments. Anschließend wählt der Kunde, ob es sich bei dem Dokument um ein Kundendokument oder ein Immobiliendokument handelt. Hier wird von einem Immobiliendokument ausgegangen, da dies der kompliziertere Fall ist. Als nächstes wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8), um dem Kunden eine Liste von Immobilien anzeigen zu können. Nachdem der Kunde die gewünschte Immobilie gewählt hat, lädt er das Dokument hoch. Danach gibt er deskriptive Daten zum Dokument (Dokumententitel, Kommentar,...) ein. Hiernach wird bei der Schlüsselverwaltung ein neuer individueller Datensatzschlüssel angefordert. Nachdem die Schlüsselverwaltung ein neues Schlüsselpaar generiert hat, wird der Public-Key dieses Schlüsselpaares retourniert. Anschließend wird bei der Schlüsselverwaltung der Public-Key des Backup-Schlüsselpaares angefordert. Dazu wird zuerst eine Liste von Schlüssel-IDs des Kunden angefordert, welche den Typ „backup“ haben. Danach wird der Schlüssel selbst angefordert. Sobald der Schlüssel retourniert wurde, wird der neue Dokumentendatensatz mit einem neuen Session-Key verschlüsselt. Anschließend wird dieser Session-Key mit dem Public-Key des neuen individuellen Datensatzschlüsselpaares und dem Public-Key des Backup-Schlüsselpaares verschlüsselt.

Im Anschluss wird beim Datenspeicher eine neue Datensatz-ID und anschließend das Lock des Kunden angefordert. Nach dem Erhalt des Lockstrings wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8) und die Metadaten des neuen Dokumentendatensatzes eingetragen. Als nächstes wird der Public-Key des Metadata-Schlüsselpaares angefordert (wiederum zwei Requests: einer für eine Liste von Schlüssel-IDs des Kunden und einer für den Schlüssel selbst). Anschließend wird der MetadataRoot-Datensatz mit einem neu generierten Session-Key verschlüsselt und dieser Session-Key mithilfe des Public-Key des Metadata-Schlüsselpaares und mit dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Abschießend werden der neue Dokumentendatensatz und der aktualisierte MetadataRoot-Datensatz im Datenspeicher gesichert und das Lock des Kunden gelöst.

Systemprozess: 1.8 Upload eines Dokuments durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
Kurzbeschreibung: Der Kunde wählt den Menüpunkt zum Upload eines Dokuments. Danach wählt der Kunde, ob das Dokument dem Kunden direkt zuzuordnen ist oder einer seiner Immobilien. Wenn das Dokument zu einer Immobilie gehört, wird der MetadatRoot-Datansatz gelesen und der Kunde wählt die entsprechende Immobilie aus. Danach wird das Dokument hochgeladen. Anschließend fordert der Kunde den Public-Key des Backupschlüsselpaares und einen neuen individuellen Datensatzschlüssel bei der Schlüsselverwaltung an. Nachdem der Datensatz verschlüsselt wurde und eine neue ID beim Datenspeicher angefordert wurde, wird das Kundenlock geholt und der MetadataRoot-Datensatz gelesen. Danach wird in diesen das neue Dokument eingetragen und der MetadataRoot-Datensatz wieder verschlüsselt. Schlussendlich werden alle Daten im Datenspeicher gesichert und das Lock gelöst.

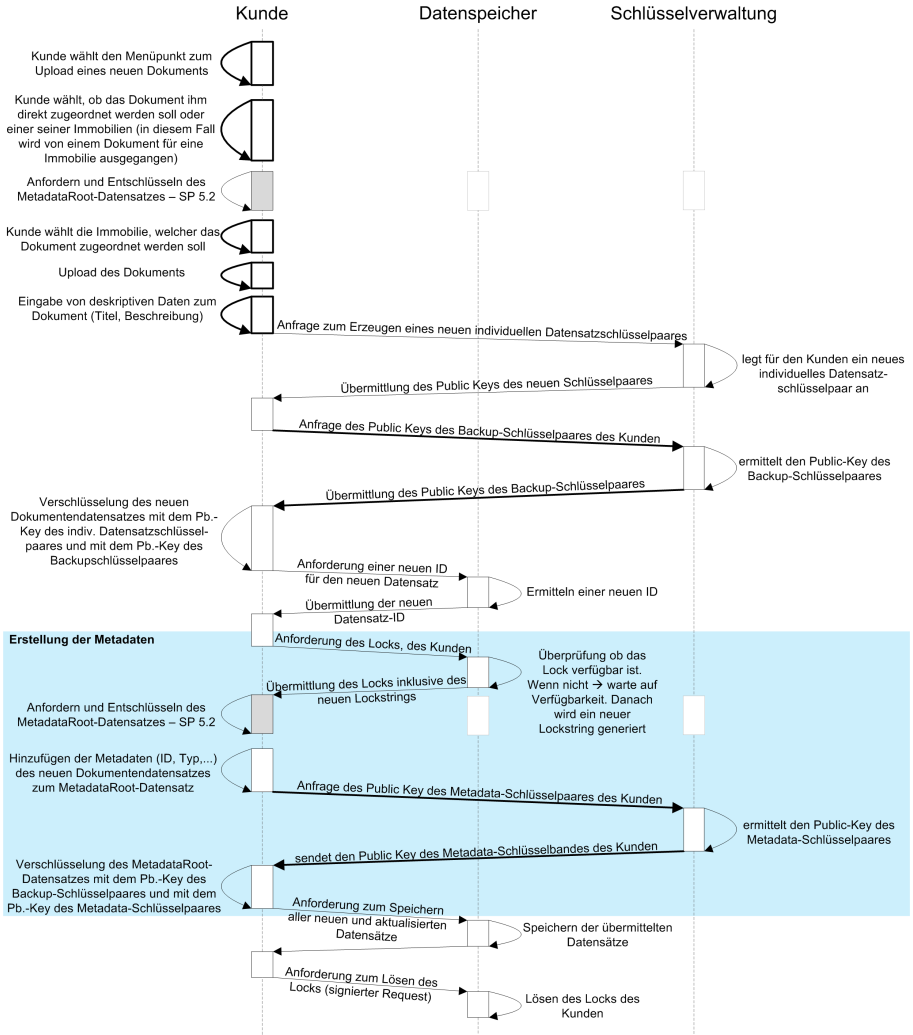


Abbildung A.10: Systemprozess 1.8 Upload eines Dokuments durch den Kunden

A.5.5 Datenabfrage

Systemprozess 2.1 Abfrage von Daten durch den Kunden

Systemprozess: 2.1 Abfrage von Daten durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
Kurzbeschreibung: Nachdem der Kunde den Menüpunkt zur Abfrage seiner Daten gewählt hat, wird zuerst der MetadataRoot-Datensatz angefordert und entschlüsselt. Anschließend werden alle Metadata-Datensätze, auf welche der MetadataRoot-Datensatz verweist, geladen und ebenfalls entschlüsselt. Danach wählt der Kunde die gewünschten Daten aus. Nachdem der Private-Key des Backup-Schlüsselpaares bei der Schlüsselverwaltung angefordert wurde, werden die ausgewählten Daten beim Datenspeicher angefordert. Anschließend werden diese mit dem Private-Key des Backup-Schlüsselpaares entschlüsselt und sind somit für den Kunden einsehbar.

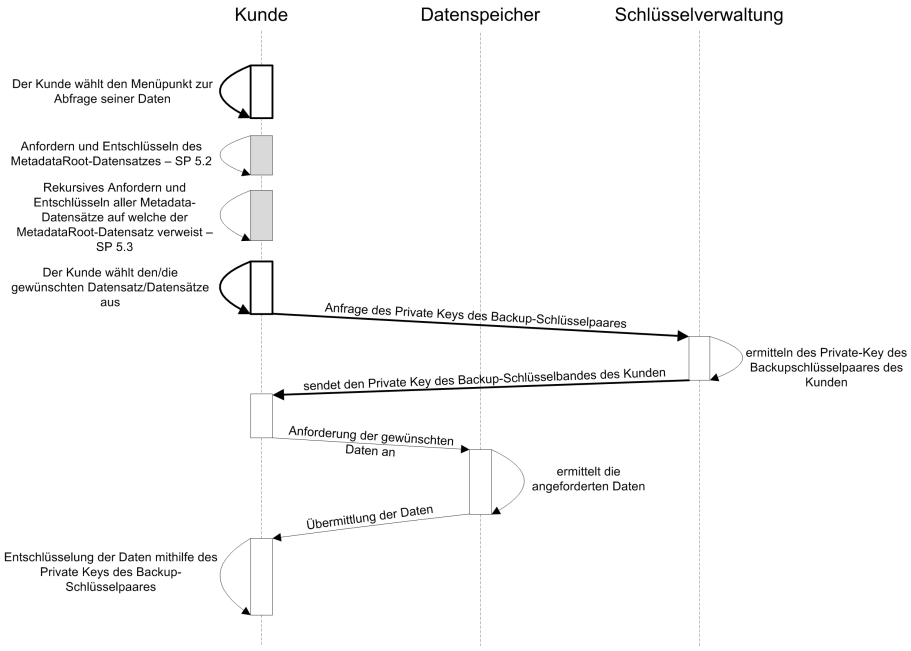


Abbildung A.11: Systemprozess 2.1 Abfrage von Daten durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)

Beschreibung: Nachdem der Kunde den Menüpunkt zur Abfrage von Daten gewählt hat, wird als erstes der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Anschließend werden alle Metadata-Datensätze, auf welche der MetadataRoot-Datensatz verweist, auch geladen und entschlüsselt (siehe Systemprozess 5.3 - Abschnitt A.5.8). Dieses Nachladen von Metadata-Datensätze

passiert rekursiv, bis alle Metadaten des Kunden geladen wurden. Als nächstes wählt der Kunde jenen Datensatz aus, welchen er einsehen möchte. Danach wird der Private-Key des Backup-Schlüsselpaares angefordert. Dies geschieht, indem zuerst eine Liste von Schlüssel-IDs des Kunden mit dem Typ „backup“ und danach der Schlüssel selbst angefordert wird. Sobald der Schlüssel geliefert wurde, wird der ausgewählte Datensatz beim Datenspeicher angefordert. Nachdem der Datensatz geliefert wurde, wird dieser mit dem zuvor angeforderten Public-Key des Backup-Schlüsselpaares, entschlüsselt. Nun kann der Kunde den entschlüsselten Datensatz einsehen.

Systemprozess 2.2 Abfrage von Daten eines Kunden durch den Energieberater

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Berater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Berater hat Zugriff auf den Private-Key des Metadata-Schlüsselpaares und auf den Private-Key des Schlüssels des entsprechenden Datensatzes

Beschreibung: Nachdem der Energieberater den Menüpunkt zur Anzeige von Daten eines Kunden ausgewählt hat, wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, auf welche der Berater zugreifen darf. Dies sind alle Schlüssel, welche der Kunde für den Berater oder für alle Benutzer freigegeben hat.

Danach wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Anschließend werden alle Metadata-Datensätze, auf welche der MetadataRoot-Datensatz verweist, auch geladen und entschlüsselt (siehe Systemprozess 5.3 - Abschnitt A.5.8). Dieses Nachladen von Metadata-Datensätze passiert rekursiv, bis alle Metadaten des Kunden geladen wurden. Als nächstes wählt der Energieberater aus jenen Datensätzen, welche er entschlüsseln kann, die gewünschten Datensätze aus. Als nächstes werden diese Datensätze beim Datenspeicher angefordert. Sobald diese Datensätze geliefert wurden, werden die zur Entschlüsselung notwendigen Private-Keys ausgewählt und bei der Schlüsselverwaltung angefordert. Die Schlüsselverwaltung prüft, ob der Berater wirklich auf die angeforderten Schlüssel zugreifen darf. Nach erfolgreicher Prüfung werden die gewünschten Schlüssel retourniert. Danach können die gewünschten Daten mit den soeben erhaltenen Private-Keys entschlüsselt werden. Somit kann der Berater nun die gewünschten Datensätze einsehen.

A.5.6 Datenanforderung und -freigabe

Systemprozess 3.1 Anforderung eines Dokuments durch den Energieberater

beteiligte Parteien: Energieberater, Datenspeicher, Schlüsselverwaltung

Systemprozess: 2.2 Abfrage von Daten eines Kunden durch den Energieberater

beteiligte Parteien: Energieberater, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Energieberater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); der Energieberater hat Zugriff auf den Metadata-Schlüssel und auf den Schlüssel des entsprechenden Datensatzes
Kurzbeschreibung: Als erstes fordert der Energieberater bei der Schlüsselverwaltung eine Liste jener Schlüssel an, welche ihm ein Kunde zur Verfügung gestellt hat. Danach wird der MetadataRoot-Datensatz des Kunden angefordert und entschlüsselt. Anschließend werden alle Metadata-Datensätze, auf welche der MetadataRoot-Datensatz verweist, geladen und ebenfalls entschlüsselt. Danach wählt der Energieberater aus jenen Datensätzen, welche er mit den zur Verfügung steheneden Schlüsseln entschlüsseln kann, die gewünschten Datensätze aus und fordert diese beim Datenspeicher an. Anschließend fordert er bei der Schlüsselverwaltung die zur Entschlüsselung notwendigen Schlüssel an und entschlüsselt die Datensätze. Der Energieberater kann nun die Datensätze im Klartext einsehen.

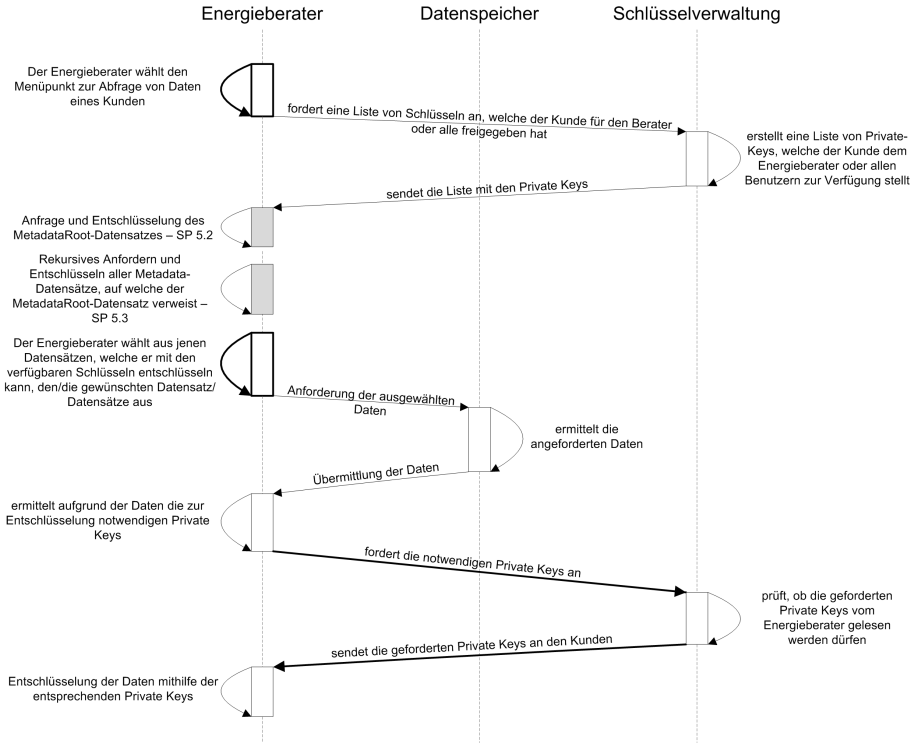


Abbildung A.12: Systemprozess 2.2 Abfrage von Daten eines Kunden durch den Energieberater

Systemprozess: 3.1 Anforderung eines Dokuments durch den Energieberater

beteiligte Parteien: Energieberater, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Energieberater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar) und hat Zugriff auf den Private-Key des Metadata-Schlüssels und auf den Private-Key der Immobilie, für welche das Dokument angefordert wird.
Kurzbeschreibung: Nachdem der Energieberater den Menüpunkt zum Anfordern von Kundenschlüsseln gewählt und den entsprechenden Kunden ausgewählt hat, wird eine Liste von Schlüsseln des Kunden angefordert, auf welche der Berater schon Zugriff hat. Danach wird eine Liste von Schlüsseln angefordert, welche der Berater bei diesem Kunden schon angefordert hat. Anschließend wird der MetadataRoot-Datensatz angefordert und entschlüsselt. Danach wählt der Kunde das gewünschte Dokument aus. Danach wird die Schlüsselanforderung in der Schlüsselverwaltung gespeichert.

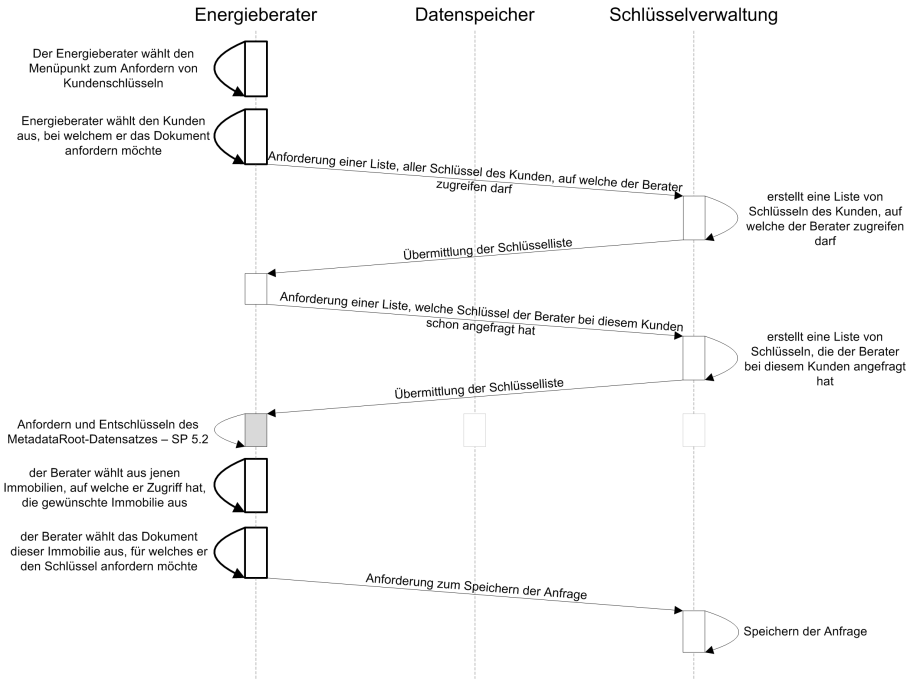


Abbildung A.13: Systemprozess 3.1 Anforderung eines Dokuments durch den Energieberater

Voraussetzungen:

- der Energieberater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Energieberater hat Zugriff auf den Private-Key des Metadata-Schlüssel und auf den Private-Key der Immobilie, für welche das Dokument angefordert wird

Beschreibung: Nachdem der Energieberater den Menüpunkt zur Anforderung von Dokumenten gewählt hat, wählt dieser den Kunden, bei welchem Daten angefordert werden sollen. Danach wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, auf welche der Berater schon zugreifen darf. Das sind all jene Schlüssel, welche der Kunde dem Berater oder allen Benutzer freigegeben hat.

Anschließend wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, welche der Berater von diesem Kunden schon angefordert hat. Sobald diese Liste zurückgeliefert wurde, wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Nach dem Entschlüsseln des MetadataRoot-Datensatzes kann dem Kunden eine Liste von Immobilien angezeigt werden, auf welche der Berater schon zugreifen darf (er hat also schon Zugriff auf die Datensatzschlüssel der Immobiliendatensätze). Aus dieser Immobilienliste wählt der Berater die gewünschte Immobilie aus. Anschließend wählt der Berater ein, zu der Immobilie gehörendes, Dokument aus, für welches er den Schlüssel noch nicht angefordert hat und welches er noch nicht entschlüsseln kann (er hat also noch keinen Zugriff auf den Datensatzschlüssel). Abschließend wird die Anfrage des individuellen Datensatzschlüssels des ausgewählten Dokuments in der Schlüsselverwaltung gespeichert.

Systemprozess 3.2 Anforderung von Messdaten durch den Energieberater

beteiligte Parteien: Energieberater, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Energieberater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Energieberater hat Zugriff auf den Private-Key des Metadata-Schlüssels und auf den Private-Key der Immobilie, für welche Messdaten angefordert werden

Beschreibung: Nachdem der Energieberater den Menüpunkt zur Anforderung von Messdaten gewählt hat, wählt er den Kunden, bei welchem Messdatend angefordert werden sollen. Danach wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, auf welche der Berater schon zugreifen darf. Das sind all jene Schlüssel, welche der Kunde dem Berater oder allen Benutzern freigegeben hat.

Anschließend wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs angefordert, welche der Berater von diesem Kunden schon angefordert hat. Sobald diese Liste zurückgeliefert wurde, wird der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Nach dem Entschlüsseln des

Systemprozess: 3.2 Anforderung von Messdaten durch den Energieberater

beteiligte Parteien: Energieberater, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Energieberater ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar) und hat Zugriff auf den Private-Key des Metadata-Schlüssels und auf den Private-Key der Immobilie, für welche Messdaten angefordert werden
Kurzbeschreibung: Nachdem der Energieberater den Menüpunkt zum Anfordern von Kundenschlüsseln gewählt und den entsprechenden Kunden ausgewählt hat, wird eine Liste von Schlüsseln des Kunden angefordert, auf welche der Berater schon Zugriff hat. Danach wird eine Liste von Schlüsseln angefordert, welche der Berater bei diesem Kunden schon angefordert hat. Anschließend wird der MetadataRoot-Datensatz angefordert und entschlüsselt. Danach wählt der Kunde die gewünschte Immobilie aus. Danach wird zuerst der Private-Key des Immobiliendatensatzes und der Immobiliendatensatz selbst angefordert und die Schlüsselband-ID gefiltert. Anschließend wählt der Berater den gewünschten Zeitraum. Danach wird eine Liste von Schlüssel-IDs des entsprechenden Schlüsselbandes angefordert und anschließend die Schlüsselanforderung in der Schlüsselverwaltung gespeichert.

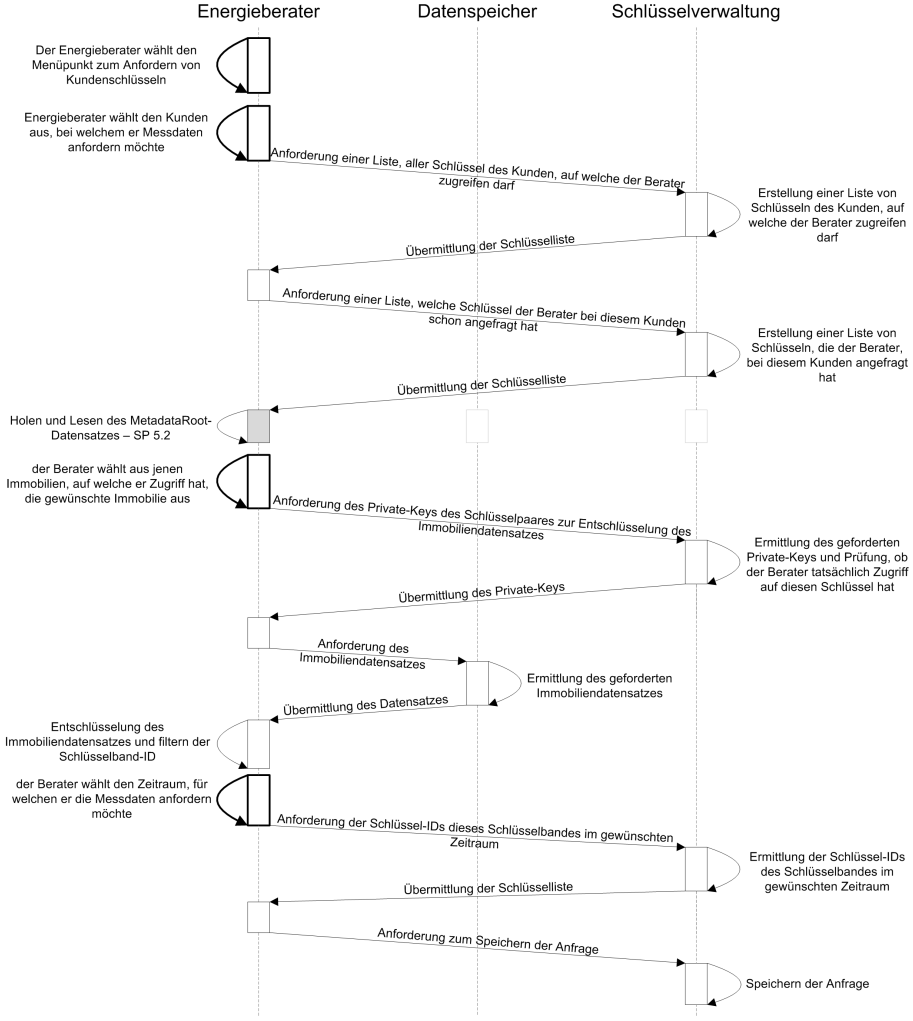


Abbildung A.14: Systemprozess 3.2 Anforderung von Messdaten durch den Energieberater

MetadataRoot-Datensatzes kann dem Kunden eine Liste von Immobilien angezeigt werden, auf welche der Berater schon zugreifen darf (er hat also schon Zugriff auf die Datensatzschlüssel der Immobiliendatensätze). Aus dieser Immobilienliste wählt der Berater die Immobilie aus, für welche er Messdaten anfordern möchte. Anschließend wird bei der Schlüsselverwaltung der Private-Key des individuellen Datensatzschlüssels der Immobilie angefordert. Nachdem die Schlüsselverwaltung überprüft hat, ob der Kunde auf den Schlüssel zugreifen darf, wird der geforderte Private-Key retourniert. Danach wird der Immobiliendatensatz beim Datenspeicher angefordert und anschließend mit dem zuvor angeforderten Private-Key entschlüsselt und die Schlüsselband-ID gefiltert.

Als nächstes wird vom Energieberater der Zeitraum auswählt, für welchen die Messdaten (und somit die entsprechenden Wochenschlüssel) angefordert werden sollen. Danach wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs des gewählten Schlüsselbandes im gewünschten Zeitraum angefordert und abschließend die Anfrage dieser Schlüssel in der Schlüsselverwaltung gespeichert.

Systemprozess 3.3 Freigabe individuellen Datensatzes eines Kunden für einen Energieberater

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)

Beschreibung: Als erstes wählt der Kunde den Menüpunkt zur Freigabe von Daten und anschließend den gewünschten Berater, dem er die Daten freigeben möchte. Anschließend wird sowohl die Liste der Schlüssel-IDs des Kunden, auf welche der Berater zugreifen darf, als auch eine Liste der von diesem Berater angeforderten Schlüssel bei der Schlüsselverwaltung angefordert. Anschließend wird der MetadataRoot-Datensatz geladen und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Zu diesem Zeitpunkt kann dem Benutzer angezeigt werden, welche Datensätze der Berater schon entschlüsseln kann und für welche Datensätze der Berater Schlüssel angefordert hat.

Als nächstes wählt der Kunde den Datensatz (Immobiliendatensatz, Kundeninformationsdatensatz oder Dokumentendatensatz) aus, für welchen er den Schlüssel freigeben möchte. Danach wird überprüft, ob der Kunde schon Zugriff auf den Private-Key des Metadata-Schlüsselpaares hat. Sollte dies nicht der Fall sein so wird dieser Schlüssel ebenfalls freigegeben. Abschließend wird die Freigabe des/der Schlüssel in der Schlüsselverwaltung gespeichert.

Systemprozess 3.4 Freigabe von Messdaten eines Kunden für einen Energieberater

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)

Systemprozess: 3.3 Freigabe eines individuellen Datensatzes eines Kunden für einen Energieberater

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
Kurzbeschreibung: Der Kunde wählt den Menüpunkt zur Datenfreigabe und wählt den gewünschten Berater aus. Anschließend fordert er eine Liste seiner Schlüssel, eine Liste jener Schlüssel, auf welche der Berater Zugriff hat, und eine Liste von Schlüsseln, welche der Berater angefordert hat, bei der Schlüsselverwaltung an. Danach wird der MetadataRoot-Datensatz geholt und entschlüsselt und der Kunde wählt den freizugebenden Datensatz. Falls der Kunde noch keinen Zugriff auf den Metadata-Schlüssel hat, wird dieser ebenfalls freigegeben. Abschließend wird in der Schlüsselverwaltung die Berechtigung des Schlüsselzugriffs durch den gewählten Berater gespeichert.

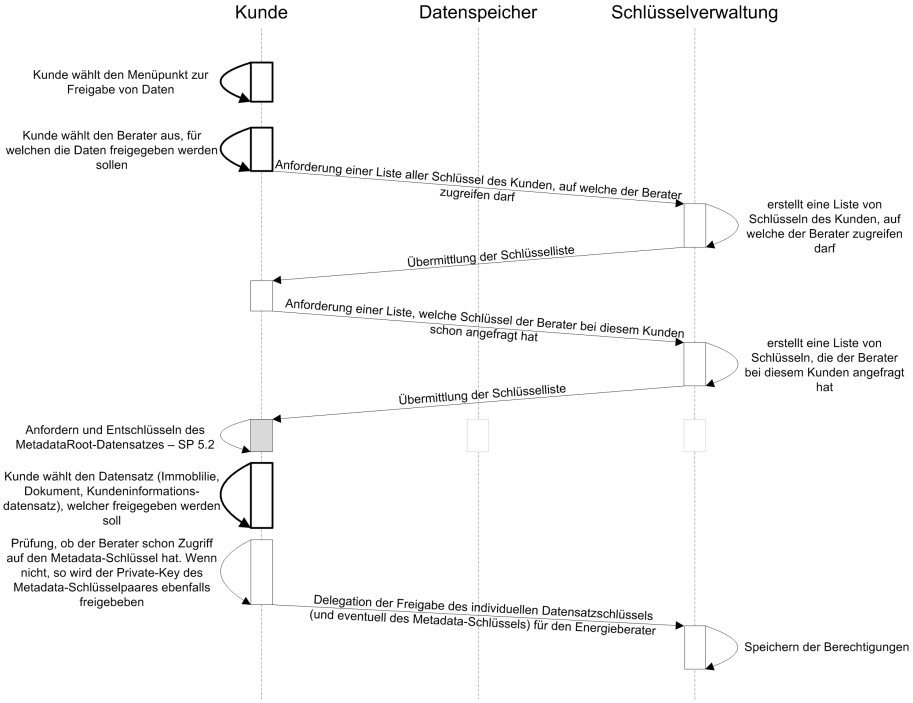


Abbildung A.15: Systemprozess 3.3 Freigabe eines individuellen Datensatzes eines Kunden für einen Energieberater

Systemprozess: 3.4 Freigabe von Messdaten eines Kunden für einen Energieberater (Teil 1)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); der gewünschte Berater hat bereits Zugriff auf den Datensatzschlüssel der entsprechenden Immobilie
Kurzbeschreibung: Der Kunde wählt den Menüpunkt zur Datenfreigabe und wählt den gewünschten Berater aus. Anschließend fordert er eine Liste seiner Schlüssel, eine Liste jener Schlüssel, auf welche der Berater Zugriff hat, und eine Liste von Schlüsseln, welche der Berater angefordert hat, bei der Schlüsselverwaltung an. Danach wird der MetadataRoot-Datensatz geholt und entschlüsselt und der Kunde wählt die Immobilie, für welche Messwerte freigegeben werden sollen. Anschließend wird geprüft ob der Berater schon Zugriff auf den Immobiliendatensatzschlüssel hat (wenn nicht, wird abgebrochen) und der Private-Key des Backupschlüsselpaares angefordert. Danach wird der Immobiliendatensatz angefordert, entschlüsselt und die Schlüsselband-ID gefiltert. Anschließend wählt der Kunde den Freigabezeitraum aus. Fortsetzung in Teil 2.

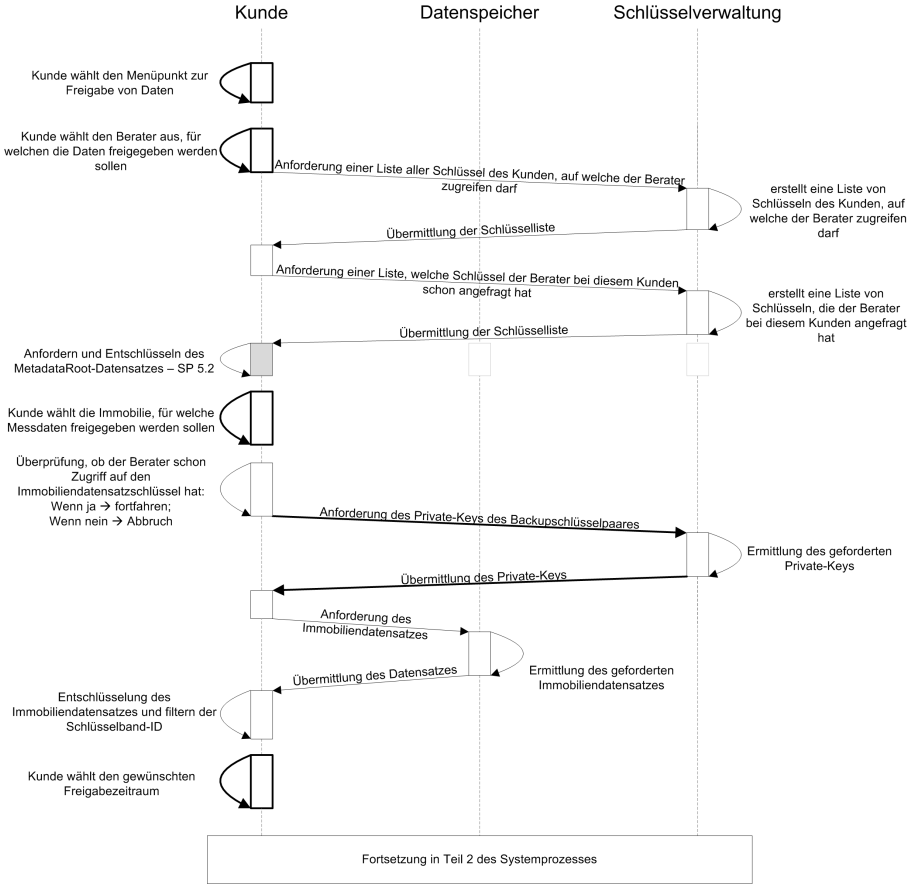


Abbildung A.16: Systemprozess 3.4 Freigabe von Messdaten eines Kunden für einen Energieberater (Teil 1)

Systemprozess: 3.4 Freigabe von Messdaten eines Kunden für einen Energieberater (Teil 2)

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); der gewünschte Berater hat bereits Zugriff auf den Datensatzschlüssel der gewünschten Immobilie; Teil 1 des Use Cases wurde erfolgreich ausgeführt
Kurzbeschreibung: Nachdem Teil 1 des Use Cases erfolgreich ausgeführt wurde, werden bei der Schlüsselverwaltung alle Schlüssel-IDs der Wochenschlüssel des Schlüsselbandes im Freigabezeitraum angefordert. Anschließend wird das Lock des Kunden geholt und alle Metadatensätze der Zähler der Immobilie geholt und entschlüsselt. Anschließend werden alle Zählerdatensätze vom Datenspeicher geholt und mit dem ersten Schlüssel des Freigabezeitraums zusätzlich verschlüsselt. Danach werden die Metadaten der Zählerdatensätze aktualisiert und im Datenspeicher gespeichert. Anschließend wird das Lock des Kunden gelöst. Abschließend wird die Freigabe der entsprechenden Wochenschlüssel in der Schlüsselverwaltung gespeichert.

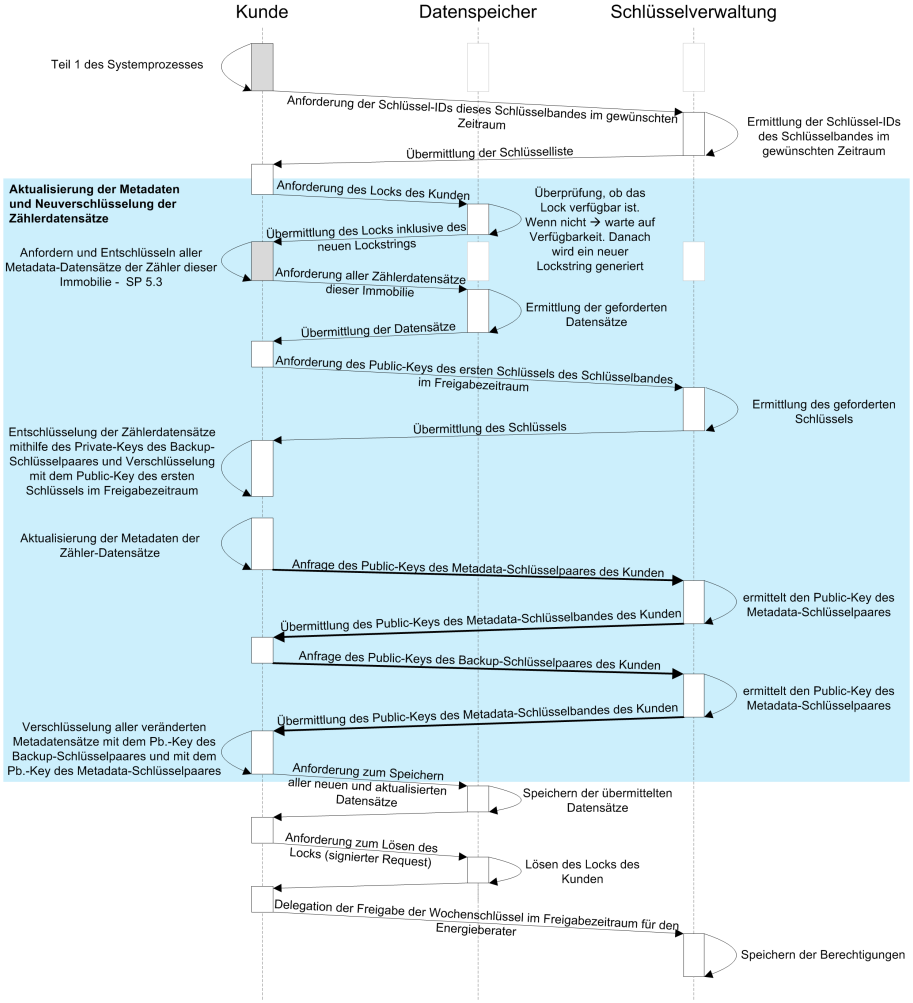


Abbildung A.17: Systemprozess 3.4 Freigabe von Messdaten eines Kunden für einen Energieberater (Teil 2)

- der gewünschte Berater hat bereits Zugriff auf den Datensatzschlüssel der entsprechenden Immobilie

Beschreibung: Als erstes wählt der Kunde den Menüpunkt zur Freigabe und den gewünschten Berater, für den die Messdaten freigegeben werden sollen. Anschließend wird sowohl die Liste der Schlüssel-IDs des Kunden, auf welche der Berater zugreifen darf, als auch eine Liste der von diesem Berater angeforderten Schlüssel bei der Schlüsselverwaltung angefordert. Anschließend wird der MetadataRoot-Datensatz geladen und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Nun kann dem Kunden eine Liste seiner Immobilien angezeigt werde, aus welcher er die gewünschte Immobilie auswählt. Danach wird überprüft, ob der ausgewählte Energieberater schon Zugriff auf den Private-Key des Immobiliendatensatzschlüsselspaares hat. Sollte dies nicht der Fall sein, wird der Systemprozess abgebrochen. Andernfalls wird als nächstes der Private-Key des Backup-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Diese Anforderung besteht aus der Anforderung einer Liste von Schlüssel-IDs des Kunden, welche den Typ „backup“ haben und der Anforderung des Schlüssels selbst. Nachdem der Schlüssel geliefert wurde, wird beim Datenspeicher der Immobiliendatensatz angefordert und entschlüsselt, um die Schlüsselband-ID zu filtern. Nach diesem Schritt wählt der Kunde den Zeitraum, für welchen er die Messdaten dieser Immobilie freigeben möchte. Danach wird bei der Schlüsselverwaltung eine Liste der Schlüssel-IDs des, zu der Immobilie gehörenden Schlüsselbandes im gewählten Zeitraum angefragt.

Als nächstes werden alle zu der Immobilie gehörenden Zählerdatensätze mit einem der neu freizugebenden Schlüssel verschlüsselt. Dies stellt sicher, dass der Berater auf jeden Fall die notwendigen Zählerdatensätze entschlüsseln und somit einsehen kann. Für die Neuverschlüsselung wird als erstes das Lock des Kunden beim Datenspeicher angefordert. Sobald das Lock verfügbar ist, wird ein neuer Lockstring übermittelt. Anschließend werden alle Metadata-Datensätze der entsprechenden Zähler geladen und entschlüsselt (siehe UseCase 5.3 - Abschnitt A.5.8). Anschließend werden alle Zählerdatensätze beim Datenspeicher angefordert. Als nächstes wird der Public-Key des ersten Schlüssels des Schlüsselbandes im Freigabezeitraum bei der Schlüsselverwaltung angefordert. Nun werden die Session-Keys der Zählerdatensätze mit dem zuvor angeforderten Private-Key des Backup-Schlüsselpaares entschlüsselt und mit dem soeben angeforderten Public-Key des Wochenschlüssels neu zusätzlich verschlüsselt. Anschließend werden die entsprechenden Metadata-Datensätze zu den Zählerdatensätzen aktualisiert. Danach wird der Public-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert (wiederum zwei Requests: ein Request für die Liste der Schlüssel-IDs des Kunden mit dem Typ „metadata“ und ein Request für den Schlüssel selbst). Anschließend wird der Public-Key des Backup-Schlüsselpaares angefordert (2 Requests). Nachdem die Schlüssel geliefert wurden, werden die aktualisierten Metadata-Datensätze der Zähler mit dem Public-Key des Backup-Schlüsselpaares und dem Public-Key des Metadata-Schlüsselpaares verschlüsselt. Anschließend werden die Metadata-Datensätze der Zähler und die neu verschlüsselten Zählerdatensätze im Datenspeicher aktualisiert und das Lock des Kunden gelöst. Abschließend wird die Freigabe der Wochenschlüssel in der Schlüsselverwaltung gespeichert.

A.5.7 Editieren von Datenobjekten

Systemprozess 4.1 Editieren eines Datenobjekts durch den Kunden

Systemprozess: 4.1 Editieren eines Datenobjekts durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung
Voraussetzung: Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar); der Kunde hat zuvor einen Datensatz mithilfe von Systemprozess 2.1 „Abfrage von Daten durch den Kunden“ gelesen
Kurzbeschreibung: Nachdem der Kunde mittels Systemprozess 2.1 den zu editierenden Datensatz gelesen hat, kann er diesen editieren. Danach werden von der Schlüsselverwaltung alle Public-Keys jener Schlüssel angefordert, mit welchen der Datensatz verschlüsselt war. Anschließend wird der Datensatz mit allen diesen Schlüsseln verschlüsselt. Danach wird das Lock des Kunden und der entsprechende Metadatensatz, in welchem der Datensatz eingetragen ist, angefordert. Dieser wird anschließend entschlüsselt, aktualisiert und wieder mit dem Public-Key des Metadata-Schlüsselpaares verschlüsselt. Abschließend werden die Daten im Datenspeicher gesichert und das Kundenlock gelöst.

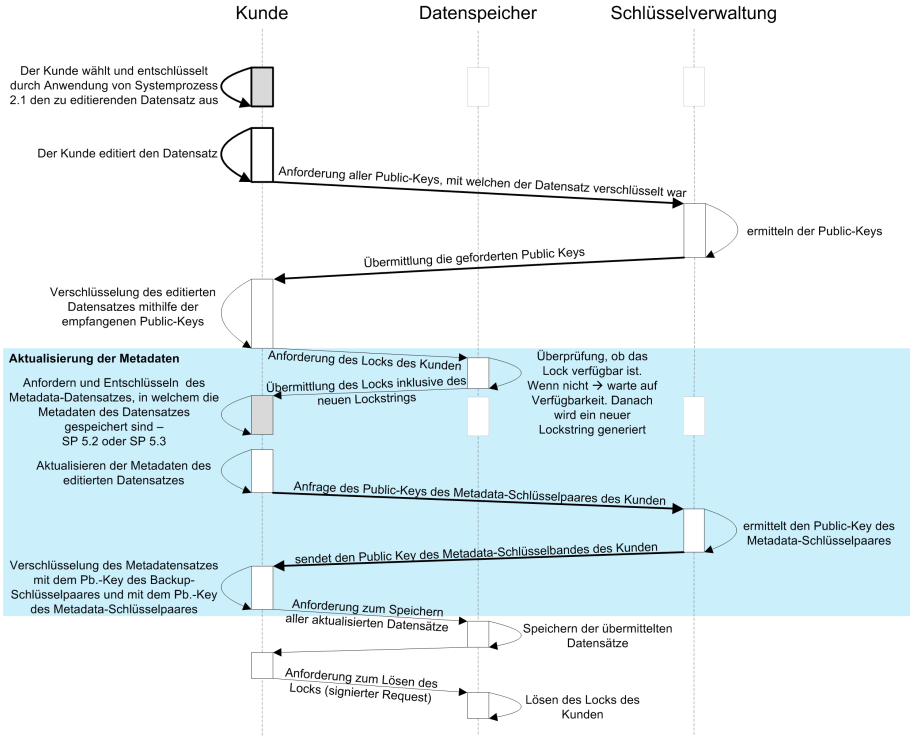


Abbildung A.18: Systemprozess 4.1 Editieren eines Datenobjekts durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Kunde hat zuvor einen Datensatz mithilfe von Systemprozess 2.1 „Abfrage von Daten durch den Kunden“ (Abschnitt A.5.5) gelesen

Beschreibung: Nachdem der Kunde durch Anwendung von Systemprozess 2.1 (Abschnitt A.5.5) den zu editierenden Datensatz angefordert und entschlüsselt hat, kann der Kunde den Datensatz editieren[1]. Nachdem der Kunde den Datensatz editiert hat, werden alle Public-Keys jener Schlüssel, mit denen der Datensatz vor dem Editieren verschlüsselt war, bei der Schlüsselverwaltung angefordert. Dazu wird für jeden Schlüssel ein Request an die Schlüsselverwaltung gesendet. Nachdem alle Schlüssel geladen wurden, wird ein neuer Session-Key erzeugt und der Datensatz damit verschlüsselt. Anschließend wird der Session-Key mithilfe der soeben geladenen Session-Keys verschlüsselt.

Danach wird das Lock des Kunden beim Datenspeicher angefordert. Sobald das Lock frei ist, wird der Lockstring retourniert. Danach wird jener Metadata-Datensatz aus dem Datenspeicher angefordert und entschlüsselt, in welchem die Metadaten für den editierten Datensatz gespeichert sind (siehe Systemprozess 5.2 bzw. 5.3 - Abschnitt A.5.8 bzw. Abschnitt A.5.8). Anschließend werden die Metadaten des Datensatzes im soeben angeforderten Metadata-Datensatz aktualisiert und der Public-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Hierzu wird zuerst eine Liste von Schlüssel-IDs des Kunden mit dem Typ „metadata“ und danach der Schlüssel selbst angefordert. Sobald die Schlüssel eingetroffen sind, wird der Metadata-Datensatz mit dem Public-Key des Metadata-Schlüsselpaares und dem Public-Key des Backup-Schlüsselpaares verschlüsselt. Danach werden alle aktualisierten Datensätze im Datenspeicher aktualisiert und abschließend das Lock des Kunden gelöst.

A.5.8 Sonstige Systemprozesse

Systemprozess 5.1 Überprüfung der Signatur eines Requests einer automatischen Datenquelle durch den Datenspeicher

beteiligte Parteien: Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- die Schlüsselverwaltung ist erreichbar
- der Datenspeicher hat einen Request von einer automatischen Datenquelle erhalten (diese sind immer digital signiert)

Beschreibung: Nachdem der Datenspeicher einen Request von einer automatischen Datenquelle erhalten hat, fordert dieser den Public-Key des Signaturschlüsselpaares, mit welchem der Request signiert wurde, von der Schlüsselverwaltung an. Die entsprechende Schlüssel-ID wurde im Request von der automatischen Datenquelle mitgesendet. Nachdem die Schlüsselverwaltung den Public-Key zurückgeliefert hat, kann die Signatur auf Gültigkeit überprüft werden.

[1] ***Hinweis:*** Datensätze können nicht uneingeschränkt editiert werden. Beispielsweise der Typ eines Datensatzes nicht geändert werden.

Systemprozess: 5.1 Überprüfung der Signatur eines Requests einer automatischen Datenquelle durch den Datenspeicher

beteiligte Parteien: Datenspeicher, Schlüsselverwaltung
Voraussetzung: die Schlüsselverwaltung ist erreichbar; der Datenspeicher hat einen Request von einer automatischen Datenquelle erhalten (diese sind immer digital signiert)
Kurzbeschreibung: Die Datenquelle fordert bei der Schlüsselverwaltung den entsprechenden Public Key des Signaturschlüsselpaares der Datenquelle an. Danach überprüft der Datenspeicher mit diesem Schlüssel die Signatur.

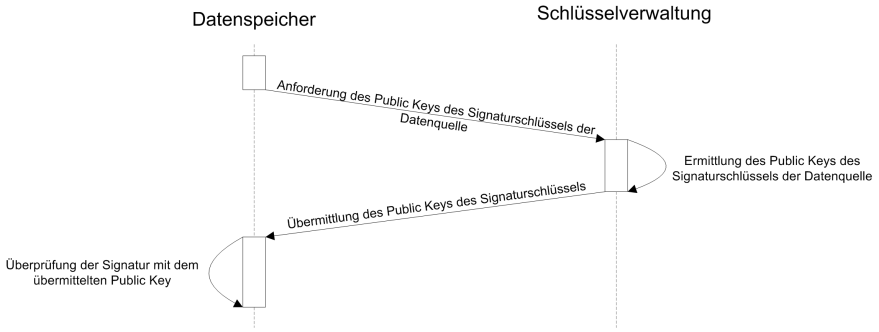


Abbildung A.19: Systemprozess 5.1 Überprüfung der Signatur eines Requests einer automatischen Datenquelle durch den Datenspeicher

Systemprozess 5.2 Anfordern und Entschlüsseln des MetadataRoot-Datensatzes durch den Kunden bzw. Berater

beteiligte Parteien: Kunde bzw. Energieberater, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- die Datenquelle ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar)
- wenn es sich beim Benutzer um einen Energieberater handelt, so muss dieser Zugriffsrechte auf den Private-Key des Metadata-Schlüsselpaares besitzen

Beschreibung: Als erstes wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs des Kunden angefordert, welche den Typ „metadata“ besitzen. Nachdem die Schlüsselverwaltung diese Liste retourniert hat, wird überprüft, ob ein solcher Schlüssel verfügbar ist. Sollte dies der Fall sein, so wird der Private-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Andernfalls wird der Systemprozess abgebrochen.

Nachdem der Schlüssel geliefert wurde, wird eine Liste von Datensätzen des Kunden mit dem Typ „metadataRoot“ beim Datenspeicher angefragt. Dabei überprüft der Datenspeicher, ob der Datensatz mit dem Schlüssel, dessen Schlüssel-ID im Request übergeben wurde, verschlüsselt ist. Sollte so ein Datensatz existieren, wird dieser im Anschluss beim Datenspeicher ebenfalls angefordert. Anschließend wird der Session-Key des Datensatzes mithilfe des Private-Keys des Metadata-Schlüsselpaares ent-

Systemprozess: 5.2 Anfordern und Entschlüsseln des MetadataRoot-Datensatzes durch den Kunden bzw. Berater

beteiligte Parteien: Kunde bzw. Energieberater, Datenspeicher, Schlüsselverwaltung
Voraussetzung: der Benutzer ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar); wenn es sich beim Benutzer um einen Energieberater handelt, so muss dieser Zugriff auf den Private-Key des Metadata-Schlüsselpaares besitzen
Kurzbeschreibung: Zuerst wird bei der Schlüsselverwaltung eine Liste von Metadaten-Schlüssel angefordert. Danach fordert der Benutzer den Datensatz mit dem Typ „MetadataRoot“ beim Datenspeicher an. Anschließend fordert der Benutzer den Private Key des Metadata-Schlüsselpaares an, mit welchem er danach den Datensatz entschlüsselt.

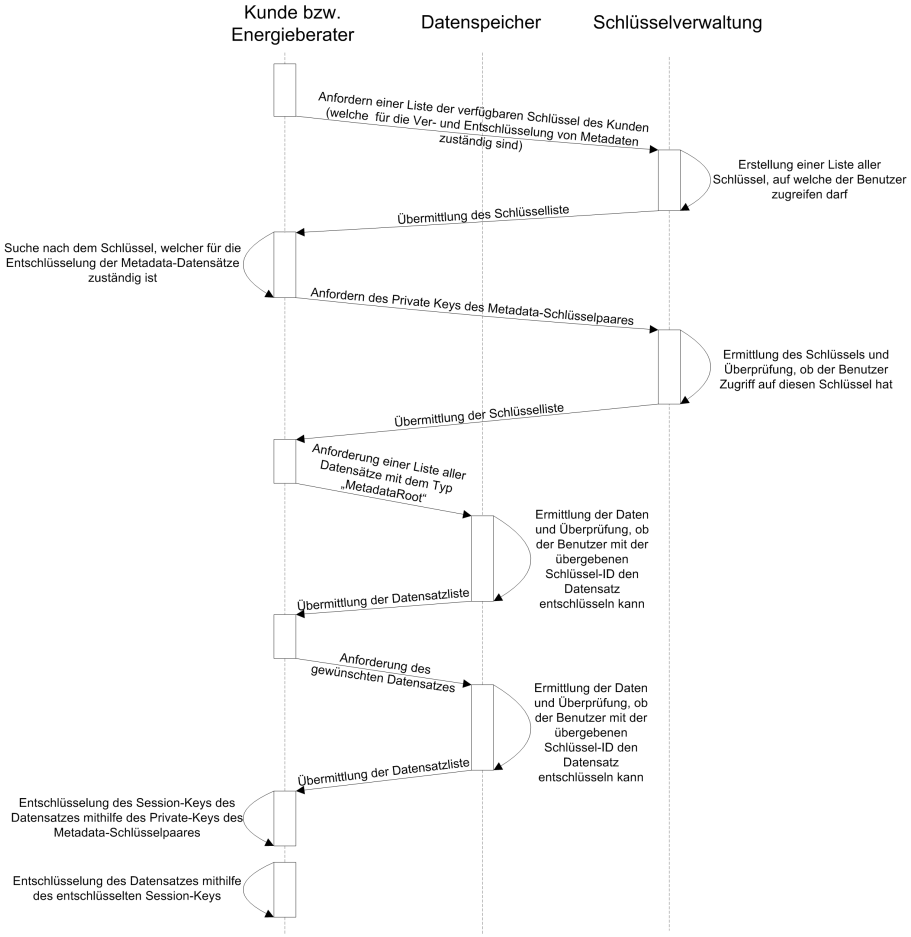


Abbildung A.20: Systemprozess 5.2 Anfordern und Entschlüsseln des MetadataRoot-Datensatzes durch den Kunden bzw. Berater

schlüsselt. Abschließend wird der MetadataRoot-Datensatz mithilfe des soeben entschlüsselten Session-Keys ebenfalls entschlüsselt.

Systemprozess 5.3 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch den Kunden bzw. Berater

Systemprozess: 5.3 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch den Kunden bzw. Berater

beteiligte Parteien: Kunde bzw. Energieberater, Datenspeicher, Schlüsselverwaltung
Voraussetzung: der Benutzer ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar); wenn es sich beim Benutzer um einen Energieberater handelt, muss dieser Zugriff auf den Private Key des Metadata-Schlüsselpaares besitzen; dem Benutzer ist die ID des Datensatzes bekannt
Kurzbeschreibung: Der Benutzer fragt zuerst die Schlüsselverwaltung nach eine Liste von Metadaten-Schlüsseln. Danach fordert der Benutzer den Datensatz beim Datenspeicher an. Anschließend fordert der Benutzer den Private Key des Metadata-Schlüsselpaares an, mit welchem er danach den Datensatz entschlüsselt.

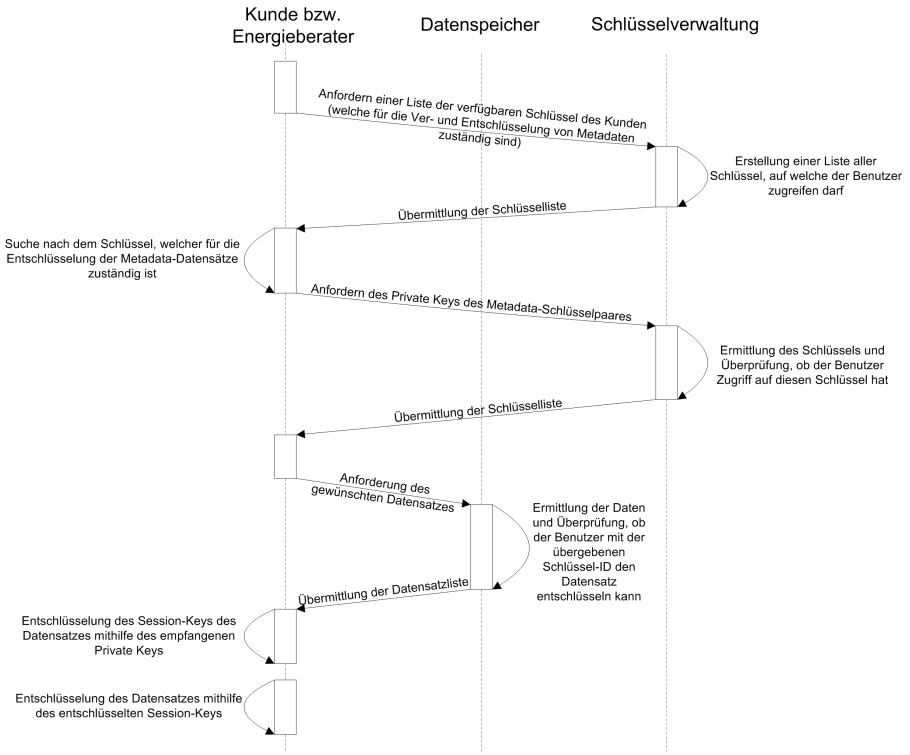


Abbildung A.21: Systemprozess 5.3 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch den Kunden bzw. Berater

beteiligte Parteien: Kunde bzw. Energieberater, Datenspeicher, Schlüsselverwaltung

Voraussetzungen:

- der Benutzer ist online (die Schlüsselverwaltung und der Datenspeicher sind erreichbar)

- wenn es sich beim Benutzer um einen Energieberater handelt, so muss dieser Zugriff auf den Private Key des Metadata-Schlüsselpaares besitzen
- dem Benutzer ist die ID des Datensatzes bekannt

Beschreibung: Als erstes wird bei der Schlüsselverwaltung eine Liste von Schlüssel-IDs des Kunden angefordert, welche den Typ „metadata“ besitzen. Nachdem die Schlüsselverwaltung diese Liste retourniert hat, wird überprüft, ob ein solcher Schlüssel verfügbar ist. Sollte dies der Fall sein, so wird der Private-Key des Metadata-Schlüsselpaares bei der Schlüsselverwaltung angefordert. Andernfalls wird der Systemprozess abgebrochen.

Nachdem der Schlüssel geliefert wurde, wird der gewünschte Metadata-Datensatz beim Datenspeicher angefordert. Dabei überprüft der Datenspeicher, ob der Datensatz mit dem Schlüssel, dessen Schlüssel-ID im Request übergeben wurde, verschlüsselt ist. Anschließend wird der Session-Key des Datensatzes mithilfe des Private-Keys des Metadata-Schlüsselpaares entschlüsselt. Abschließend wird der Metadata-Datensatz mithilfe des soeben entschlüsselten Session-Keys ebenfalls entschlüsselt.

Systemprozess 5.4 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch eine automatische Datenquelle

Systemprozess: 5.4 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch eine automatische Datenquelle

beteiligte Parteien: automatische Datenquelle (Zähler), Datenspeicher
Voraussetzung: die Datenquelle ist online (der Datenspeicher ist erreichbar); der Datenquelle ist die ID des Datensatzes bekannt; die Datenquelle ist korrekt parametriert
Kurzbeschreibung: Die Datenquelle fordert beim Datenspeicher den Metadata-Datensatz an. Der Datenspeicher überprüft die Signatur des Requests und sendet den Metadata-Datensatz an die Datenquelle. Die Datenquelle entschlüsselt den Metadata-Datensatz

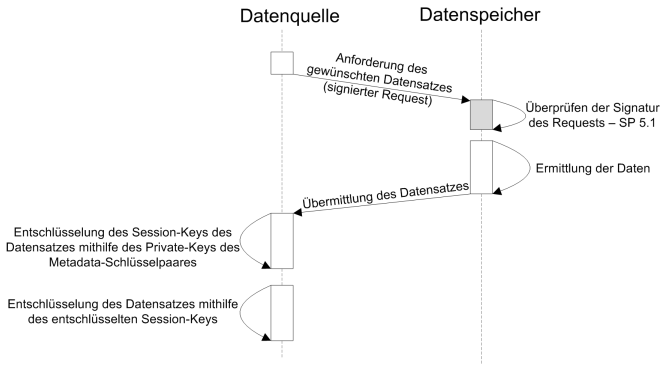


Abbildung A.22: Systemprozess 5.4 Anfordern und Entschlüsseln eines Metadata-Datensatzes durch eine automatische Datenquelle

beteiligte Parteien: automatische Datenquelle (Zähler), Datenspeicher

Voraussetzungen:

- die Datenquelle ist online (der Datenspeicher ist erreichbar)
- der Datenquelle ist die ID des Datensatzes bekannt
- die Datenquelle ist korrekt parametriert

Beschreibung: Da bei der automatischen Datenquelle der Private-Key des Metadata-Schlüsselpaares parametriert ist, braucht dieser nicht bei der Schlüsselverwaltung angefordert werden. Aus diesem Grund kann der Use-Case gleich mit der Anforderung des gewünschten Metadata-Datensatzes beginnen. Da dieser Request signiert gesendet wird, muss der Datenspeicher als erstes die Signatur des Requests auf Gültigkeit prüfen (siehe Systemprozess 5.1 - Abschnitt A.5.8).

Nachdem der Datensatz retourniert wurde, wird der Session-Key des Datensatzes mithilfe des parametrierten Private-Key des Metadata-Schlüsselpaares enschlüsselt. Abschließend wird der Metadata-Datensatz mithilfe des soeben enschlüsselten Session-Keys ebenfalls entschlüsselt.

Systemprozess 5.5 Überprüfung auf Vorhandensein eines Kundeninformationsdatensatzes durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher 1, Datenspeicher 2, Schlüsselverwaltung

Voraussetzungen:

- der Kunde ist online (beide Datenspeicher und Schlüsselverwaltung sind erreichbar)
- der Kunde hat in beiden Datenspeichern Daten abgelegt

Beschreibung: Dieser ist der einzige Systemprozess, welcher mehrere Datenspeicher benützt (und zwar all jene Datenspeicher, in welchen der Kunde Daten abgelegt hat). Bei allen anderen Systemprozessen wird immer ein bestimmter Datenspeicher benutzt, welchen der Benutzer vorher auswählt. Dieser Systemprozess benützt deswegen mehrere Datenspeicher, weil jeder Kunde nur einen einzigen Kundeninformationsdatensatz abspeichern darf. Aus diesem Grund muss vor dem Anlegen eines solchen Datensatzes geprüft werden, ob ein solcher Datensatz schon existiert.

Dazu wird als erstes eine Liste aller Schlüssel des Kunden bei der Schlüsselverwaltung angefordert. Da jeder individuelle Datensatzschlüssel einen Verweis auf den Datenspeicher, in welchem der zugehörige Datensatz gespeichert ist, enthält, können aus dieser Liste die Datenspeicher, in welchen der Kunde Daten abgelegt hat, herausgefunden werden.

Als nächstes wird von jedem dieser Datenspeicher der MetadataRoot-Datensatz angefordert und entschlüsselt (siehe Systemprozess 5.2 - Abschnitt A.5.8). Da die Metadaten des Kundeninformationsdatensatzes im MetadataRoot-Datensatz eingetragen werden, kann aus diesen Datensätzen ermittelt werden, ob schon ein Kundeninformationsdatensatz gespeichert wurde. Sobald ein Verweis auf einen Kundeninformationsdatensatz gefunden wurde, wird „gefunden“ retourniert. Wird in keinem MetadataRoot-Datensatz ein solcher Verweis gefunden, so wird „nicht gefunden“ retourniert.

Systemprozess: 5.5 Überprüfung auf Vorhandensein eines Kundeninformationsdatensatzes durch den Kunden

beteiligte Parteien: Kunde, Datenspeicher 1, Datenspeicher 2, Schlüsselverwaltung
Voraussetzung: Kunde ist online (beide Datenspeicher und Schlüsselverwaltung sind erreichbar); der Kunde hat in beiden Datenspeichern Daten abgelegt
Kurzbeschreibung: Zuerst wird bei der Schlüsselverwaltung eine Liste aller Schlüssel des Kunden angefordert. Aus dieser Liste kann ermittelt werden, in welchen Datenspeichern der Kunde Daten abgelegt hat. Danach wird für jeden dieser Datenspeicher der MetadataRoot-Datensatz angefordert, entschlüsselt und überprüft, ob ein Kundeninformationsdatensatz gefunden wurde. Wenn ein solcher gefunden wurde, wird die Suche abgebrochen und „gefunden" retourniert. Wird kein Datensatz gefunden, so wird „nicht gefunden" retourniert.

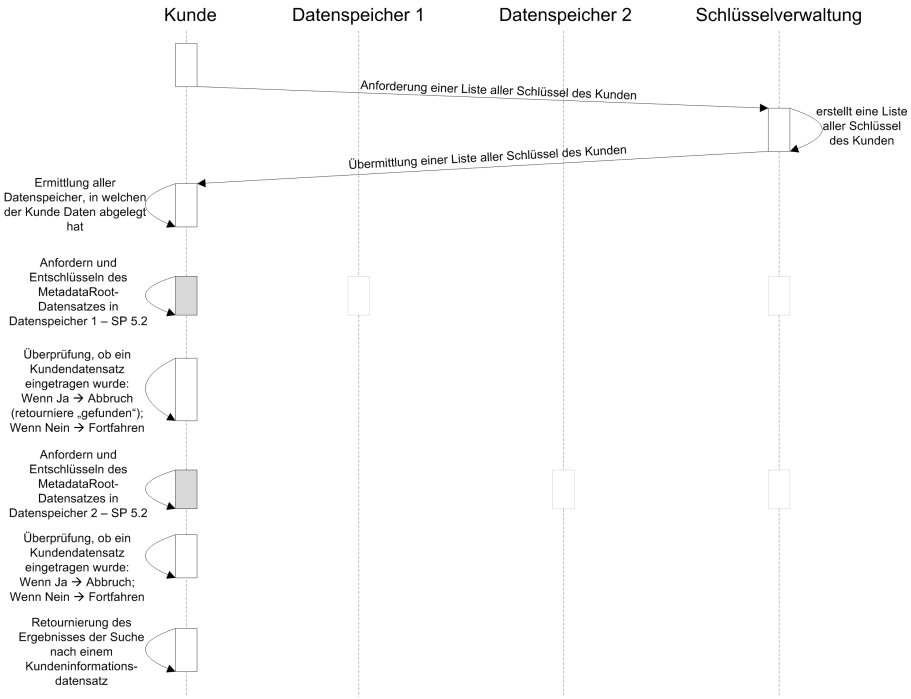


Abbildung A.23: Systemprozess 5.5 Überprüfung auf Vorhandensein eines Kundeninformationsdatensatzes durch den Kunden

A.6 Interfacebeschreibung

Dieser Abschnitt beschreibt die Schnittstelle zwischen den Clientapplikationen (Clientapplikation des Kunden, Clientapplikation des Beraters und automatische Datenquelle) und der Schlüsselverwaltung bzw. den Clientapplikationen und dem Datenspeicher. Darüber hinaus wird die Schnittstelle zwischen Datenspeicher und Schlüsselverwaltung beschrieben, welche für die Anforderung der Public-Keys der Signatur-Schlüsselpaare durch den Datenspeicher genutzt wird.

Darüber hinaus dienen die hier beschriebenen Schnittstellen als Programmier-Interface für die Web-Services der Schlüsselverwaltung und des Datenspeichers. Dabei kann jedes hier beschriebene Interface als eigenständiges Service betrachtet werden.

Zur Beschreibung der Interfaces werden alle Funktionen der jeweiligen Serviceschnittstelle vorgestellt und deren Eingangs- und Ausgangsparameter beschrieben. Es wird ausführlich erklärt, welche Aufgabe die vorgestellte Funktion erfüllt und welcher Benutzer die Funktion aufrufen darf. Sollte eine Funktion von einem unberechtigten Benutzer aufgerufen werden, so wird der Request abgelehnt und ein entsprechender Fehler retourniert. Bei vielen Funktionen werden für die Parameter eigens definierte komplexe Datentypen verwendet, welche hier auch vorgestellt und detailliert beschrieben werden.

Eine maschinenverständliche Definition der Serviceschnittstellen in Form von WSDL[2]-Dokumenten und XML-Schemas (*.xsd-Files) ist im Anhang A.7 zu finden.

A.6.1 Interface zwischen Client und Schlüsselverwaltung

Funktion *AddNewUser*

Diese Funktion wird dazu verwendet, um einen neuen Benutzer bei der Schlüsselverwaltung anzulegen. Dabei wird geprüft, ob schon Benutzer mit dem übergebenen Benutzernamen existiert. Sollte dies der Fall sein, so wird ein Fehler zurückgegeben. Sollte der Benutzername noch frei sein, so wird ein neuer Benutzer angelegt und ein neues Backup-Schlüsselpaar und ein neues Metadata-Schlüsselpaar generiert, welche allesamt in der Schlüsselverwaltung gespeichert werden.

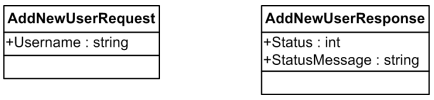


Abbildung A.24: Klassendiagramme der Klassen *AddNewUserRequest* und *AddNewUserResponse*

[2] Web Service Description Language

Request *AddNewUserRequest*

Attribut	Beschreibung
Username (string)	der Benutzername, des neu anzulegenden Benutzers

Response *AddNewUserResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *ListDataStorage*

Diese Funktion wird dazu verwendet um eine Liste von verfügbaren Datenspeichern abzufragen. Diese Funktion darf ohne Einschränkung von jedem Benutzer aufgerufen werden.

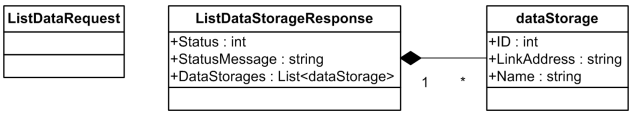


Abbildung A.25: Klassendiagramme der Klassen *ListDataStorageRequest*, *ListDataStorageResponse und dataStorage*

Request *ListDataStorageRequest*

Der Request enthält keine Attribute.

Response *ListDataStorageResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
DataStorages (List<dataStorage>)	eine Liste von Datenspeichern (inkl. der dazu notwendigen Informationen)

Zusätzlicher Typ *dataStorage*

Attribut	Beschreibung
ID (int)	eindeutige ID des Datenspeichers.
LinkAddress (string)	Dieses Attribut enthält die URL, unter welcher der Datenspeicher erreichbar ist.
Name (string)	Dieses Attribut enthält den Namen des Datenspeichers, welcher zur Anzeige für den Benutzer benutzt werden kann.

Funktion *CreateNewKeyband*

Mit dieser Funktion kann ein Kunde ein neues Schlüsselband für eine Immobilie anlegen. Diese Funktion darf nur vom Kunden selbst ausgeführt werden.

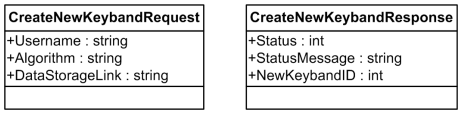


Abbildung A.26: Klassendiagramme der Klassen *CreateNewKeybandRequest* und *CreateNewKeybandResponse*

Request *CreateNewKeybandRequest*

Attribut	Beschreibung
Username (string)	Benutzername jenes Kunden, welcher ein neues Schlüsselband anlegen möchte.
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen die Schlüssel dieses Schlüsselbandes dienen sollen. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Response *CreateNewKeybandResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeybandID (int)	die eindeutige ID des neu angelegten Schlüsselbandes

Funktion *CreateNewIndividualKey*

Mit dieser Funktion kann ein Kunde ein neues individuelles Datensatz-Schlüsselpaar erzeugen. Diese Funktion darf ausschließlich vom Kunden selbst ausgeführt werden.

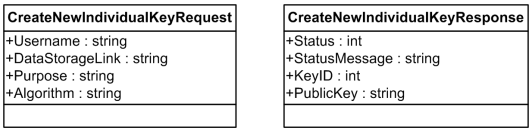


Abbildung A.27: Klassendiagramme der Klassen *CreateNewIndividualKeyRequest* und *CreateNewIndividualKeyResponse*

Request *CreateNewIndividualKeyRequest*

Attribut	Beschreibung
Username (string)	Benutzername jenes Kunden, welcher eins neues individuelles Datensatz-Schlüsselpaar anlegen möchte.
DataStorageLink (string)	URL jenes Datenspeichers, in welchem der neue Datensatz, für welchen der Schlüssel angelegt werden soll, gespeichert wird.
Purpose (string)	Art des Datensatzes, welcher mit dem neuen Datensatzschlüssel verschlüsselt wird. Momentan sind folgende Arten verfügbar: *facility* - Immobiliendatensatz *customer* - Kundeninformationsdatensatz *document* - Dokumentendatensatz
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen das Schlüsselpaar dienen soll. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Response *CreateNewIndividualKeyResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeyID (int)	die eindeutige ID des neu angelegten Schlüsselpaares
PublicKey (string)	Public-Key des neu angelegten Schlüsselpaares im XML-Format

Funktion *ListKeysOfUser*

Mit dieser Funktion kann ein Benutzer alle zugreifbaren Schlüssel eines Kunden auflisten. Diese Funktion darf von allen Benutzern aufgerufen werden.

Wenn die Funktion vom Schlüsselbesitzer (Kunde) selbst aufgerufen wird, kann dieser damit alle seine Schlüssel auflisten. Sollte die Funktion nicht vom Schlüssel besitzer aufgerufen werden, so werden lediglich jene Schlüssel aufgelistet, auf welche der Benutzer zugreifen darf (d.h. alle Private-Keys, welche der Kunde dem jeweiligen Benutzer freigegeben hat).

Der aufrufende Benutzer hat außerdem die Möglichkeit, bei der Abfrage nach der Art des Schlüssels (*purpose*) zu filtern.

Signaturschlüssel werden von dieser Funktion nicht aufgelistet.

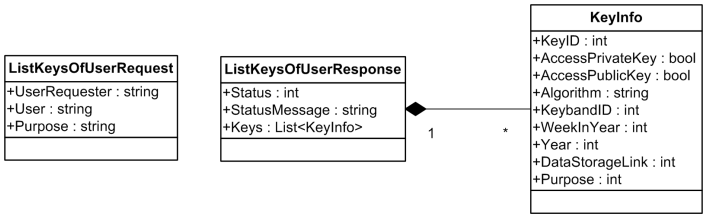


Abbildung A.28: Klassendiagramme der Klassen *ListKeysOfUserRequest*, *ListKeysOfUserResponse* und *KeyInfo*

Request *ListKeysOfUserRequest*

Attribut	**Beschreibung**
User (string)	Benutzername des Schlüsselbesitzers
UserRequester (string)	Benutzername des Benutzers, welcher die Schlüsselliste anfragt.
Purpose (string)	Art des Schlüssels. Dieses Attribut dient dazu, um nur Schlüssel einer bestimmten Art zu suchen. Momentan sind folgende Werte verfügbar: *backup* - Backup-Schlüsselpaar des Schlüsselbesitzers *metadata* - Schlüsselpaar zur Ver- und Entschlüsselung von Metadata-Datensätzen *facility* - Immobiliendatensatz-Schlüsselpaar *customer* - Kundeninformationsdatensatz-Schlüsselpaar *document* - Dokumentendatensatz-Schlüsselpaar *weeklyKey* - Wochenschlüsselpaar

Response *ListKeysOfUserResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.

Attribut	Beschreibung
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
Keys (List<KeyInfo>)	Liste der zugreifbaren Schlüssel

Zusätzlicher Typ *KeyInfo*

Attribut	Beschreibung
KeyID (int)	eindeutige Schlüssel-ID Schlüssels.
AccessPrivateKey (bool)	Gibt an, ob der Private-Key des Schlüsselpaares zugreifbar ist: *true* - Zugriff auf den Private-Key erlaubt *false* - Zugriff auf den Private-Key nicht erlaubt
AccessPublicKey (bool)	Gibt an, ob der Public-Key des Schlüsselpaares zugreifbar ist: *true* - Zugriff auf den Public-Key erlaubt *false* - Zugriff auf den Public-Key nicht erlaubt
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen das Schlüsselpaar eingesetzt wird. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus
KeybandID (int)	Schlüsselband-ID des Schlüssels (nur bei Wochenschlüsseln $Purpose = weeklyKey$, ansonsten Wert 0)
WeekInYear (int)	Kalenderwoche, zu welcher der Schlüssel gehört (nur bei Wochenschlüsseln $Purpose = weeklyKey$, ansonsten Wert 0)
Year (int)	Jahr, zu welchem der Schlüssel gehört (nur bei Wochenschlüsseln $Purpose = weeklyKey$, ansonsten Wert 0)
DataStorageLink (string)	gibt bei individuellen Datensatzschlüsseln ($Purpose = document$, $Purpose = customer$ oder $Purpose = facility$) die URL des Datenspeichers an, in welcher der zugehörige Datensatz gespeichert ist
Purpose (string)	gibt die Art des Schlüssels an: *backup* - Backup-Schlüsselpaar des Schlüsselbesitzers *metadata* - Schlüsselpaar zur Ver- und Entschlüsselung von Metadata-Datensätzen *facility* - Immobiliendatensatz-Schlüsselpaar *customer* - Kundeninformationsdatensatz-Schlüsselpaar *document* - Dokumentendatensatz-Schlüsselpaar *weeklyKey* - Wochenschlüsselpaar

Funktion *GetPublicKey*

Mit dieser Funktion kann ein Kunde den Public-Key eines seiner gespeicherten Schlüsselpaare abfragen. Diese Funktion darf ausschließlich vom Schlüsselbesitzer selbst ausgeführt werden.



Abbildung A.29: Klassendiagramme der Klassen *GetPublicKeyRequest* und *GetPublicKeyResponse*

Request *GetPublicKeyRequest*

Attribut	**Beschreibung**
User (string)	Benutzername des Schlüsselbesitzers.
KeyID (int)	eindeutige Schlüssel-ID des angeforderten Public-Keys

Response *GetPublicKeyResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
PublicKey (string)	Public-Key des angeforderten Schlüsselpaares im XML-Format
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen der Schlüssel verwendet wird. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Funktion *GetPrivateKey*

Mit dieser Funktion kann ein Benutzer den Private-Key eines Schlüsselpaares eines Kunden abfragen. Diese Funktion darf von jedem Benutzer ausgeführt werden. Der Private-Key wird allerdings nur zurückgeliefert, wenn der Benutzer, welcher den

Schlüssel anfordert, die nötigen Zugriffsrechte auf den Schlüssel besitzt. Zugriff haben neben dem Schlüsselbesitzer alle Benutzer, welchen der Schlüssel freigegeben wurde. Der Schlüssel kann vom Schlüsselbesitzer allen oder bestimmten Benutzern freigegeben werden.

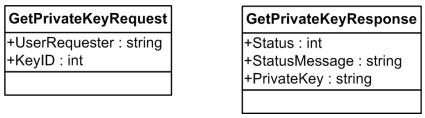


Abbildung A.30: Klassendiagramme der Klassen *GetPrivateKeyRequest* und *GetPrivateKeyResponse*

Request *GetPrivateKeyRequest*

Attribut	**Beschreibung**
UserRequester (string)	Benutzername jenes Benutzers, welcher den Schlüssel anfordert.
KeyID (int)	eindeutige Schlüssel-ID des angeforderten Private-Keys

Response *GetPrivateKeyResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt und der Schlüssel wurde erfolgreich ermittelt (der Benutzer besitzt also die nötigen Berechtigungen). $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
PrivateKey (string)	Private-Key des angeforderten Schlüsselpaares im XML-Format
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen der Schlüssel verwendet wird. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Funktion *CreateNewSignatureKey*

Diese Funktion dient zur Erzeugung eines neuen Signaturschlüsselpaares, welches der Kunde für die Parametrierung eines neuen automatischen Zählers benötigt. Diese Funktion darf nur vom Schlüsselbesitzer (Kunde) selbst ausgeführt werden. Signaturschlüssel können nicht freigegeben werden.

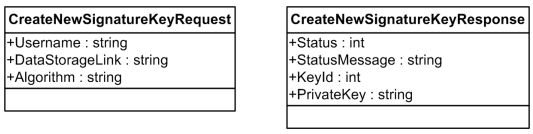


Abbildung A.31: Klassendiagramme der Klassen *CreateNewSignatureKeyRequest* und *CreateNewSignatureKeyResponse*

Request *CreateNewSignatureKeyRequest*

Attribut	**Beschreibung**
Username (string)	Benutzername des Schlüsselbsitzers
DataStorageLink	URL des Datenspeichers, in welchen der automatische Zähler, welcher diesen Signaturschlüssel verwendet, die Daten einliefert.
Algorithm (string)	Name des Signaturalgorithmus, für welchen das neue Signaturschlüsselpaar angelegt werden soll. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Response *CreateNewSignatureKeyResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeyId (int)	eindeutige ID des neu angelegten Signaturschlüsselpaares
PrivateKey (string)	Private-Key des neuen Signaturschlüssels im XML-Format

Funktion *GetPublicKeyFromKeyband*

Diese Funktion dient dazu, den Public-Key eines Wochenschlüsselpaares anzufordern. Diese Funktion darf nur vom Schlüsselbesitzer (Kunde) selbst aufgerufen werden.

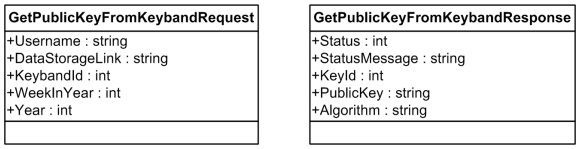


Abbildung A.32: Klassendiagramme der Klassen *GetPublicKeyFromKeybandRequest* und *GetPublicKeyFromKeybandResponse*

Request *GetPublicKeyFromKeybandRequest*

Attribut	**Beschreibung**
Username (string)	Benutzername des Schlüsselbesitzers
DataStorageLink	URL des Datenspeichers, in welchen der Zähler, welcher Schlüssel dieses Schlüsselbandes verwendet, Messwertdaten speichert.
KeybandId (int)	eindeutige ID des Schlüsselbandes, für welches der Schlüssel angefordert wird.
WeekInYear (int)	Kalenderwoche, für welche der Schlüssel angefordert wird.
Year (int)	Jahr, für welches der Schlüssel angefordert wird.

Response *GetPublicKeyFromKeybandResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeyId (int)	eindeutige ID des Wochenschlüsselpaares
PublicKey (string)	der angeforderte Public-Key im XML-Format
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen der Schlüssel vorgesehen ist. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

Funktion *GetShareRequestList*

Mit dieser Funktion erhält man eine Liste von Schlüsseln, welche ein Benutzer (Schlüsselanforderer - z.B. ein Berater) bei einem Kunden angefordert hat. Diese Funktion darf sowohl vom Schlüsselbesitzer, als auch vom Schlüsselanforderer aufgerufen werden.



Abbildung A.33: Klassendiagramme der Klassen *GetShareRequestListRequest*, *GetShareRequestListResponse* und *SharingKeyInfo*

Request *GetShareRequestListRequest*

Attribut	**Beschreibung**
UsernameKeyRequester (string)	Benutzername des Schlüsselanforderers
UsernameKeyOwner (string)	Benutzername des Schlüsselbesitzers

Response *GetShareRequestListResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
SharingKeyInfoList (List<SharingKeyInfo>)	Liste der Schlüsselinformationen

Zusätzlicher Typ *SharingKeyInfo*

Attribut	**Beschreibung**
KeyID (int)	eindeutige ID des Schlüsselpaares.

Attribut	Beschreibung
KeybandID (int)	eindeutige ID des Schlüsselbandes, zu welchem der Schlüssel gehört. Sollte der Schlüssel zu keinem Schlüsselband gehören, hat dieses Attribut den Wert 0.
WeekInYear (int)	Kalenderwoche des Schlüsselpaares. Wenn es sich bei dem Schlüsselpaar um ein Wochenschlüsselpaar handelt, so enthält dieses Attribut die Kalenderwoche des Wochenschlüssels. Andernfalls hat dieses Attribut den Wert 0.
Year (int)	Jahr des Schlüsselpaares. Wenn es sich bei dem Schlüsselpaar um ein Wochenschlüsselpaar handelt, so enthält dieses Attribut die Jahreszahl des Wochenschlüssels. Andernfalls hat dieses Attribut den Wert 0.
Purpose (string)	Art des Schlüssels. Dieses Attribut gibt den Verwendungszweck des Schlüssels an. Momentan sind folgende Werte verfügbar: *metadata* - Schlüsselpaar zur Ver- und Entschlüsselung von Metadata-Datensätzen *facility* - Immobiliendatensatz-Schlüsselpaar *customer* - Kundeninformationsdatensatz-Schlüsselpaar *document* - Dokumentendatensatz-Schlüsselpaar *weeklyKey* - Wochenschlüsselpaar

Funktion *AddShareRequest*

Mithilfe dieser Funktion kann ein Benutzer (Schlüsselanforderer - z.B. Berater) Schlüssel eines Kunden anfordern. Diese Funktion darf von allen Benutzern aufgerufen werden.

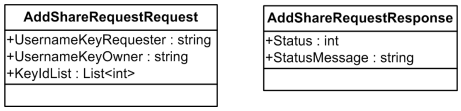


Abbildung A.34: Klassendiagramme der Klassen *AddShareRequestRequest* und *AddShareRequestResponse*

Request *AddShareRequestRequest*

Attribut	Beschreibung
UsernameKeyRequester (string)	Benutzername des Schlüsselanforderers
UsernameKeyOwner (string)	Benutzername des Schlüsselbesitzers

Attribut	Beschreibung
KeyIdList (List<int>)	Liste der Schlüssel-IDs jener Schlüssel, die angefordert werden sollen

Response *AddShareRequestResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *DeleteShareRequest*

Mit dieser Funktion kann ein Schlüsselbesitzer (Kunde) alle Schlüsselanforderungen, welche von einem bestimmten Benutzer (Schlüsselanforderer) an ihn gestellt wurden, löschen. Diese Funktion darf nur vom Schlüsselbesitzer selbst aufgerufen werden.

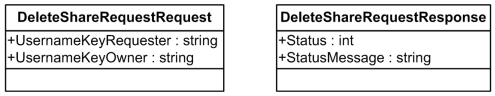


Abbildung A.35: Klassendiagramme der Klassen *DeleteShareRequestRequest* und *DeleteShareRequestResponse*

Request *DeleteShareRequestRequest*

Attribut	Beschreibung
UsernameKeyRequester (string)	Benutzername des Schlüsselanforderers, dessen Anforderungen gelöscht werden sollen.
UsernameKeyOwner (string)	Benutzername des Schlüsselbesitzers

Response *DeleteShareRequestResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *ListKeysFromKeyband*

Mithilfe dieser Funktion kann ein Benutzer eine Liste von Schlüssel-IDs eines bestimmten Schlüsselbandes in einem bestimmten Zeitraum abfragen. Dies kann dazu benutzt werden, um die anzufordernden Schlüssel-IDs, welche dieser Benutzer beim Schlüsselbesitzer anfordern möchte, herauszufinden. Diese Funktion darf von allen Benutzern ausgeführt werden.

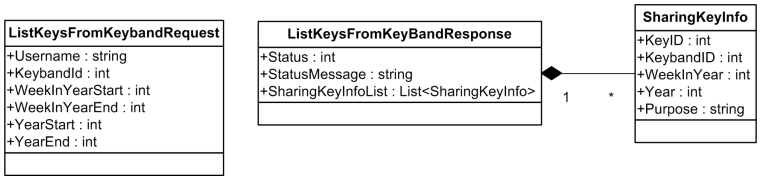


Abbildung A.36: Klassendiagramme der Klassen *ListKeysFromKeybandRequest*, *ListKeysFromKeybandResponse* und *SharingKeyInfo*

Request *ListKeysFromKeybandRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Schlüsselbesitzers
KeybandId (int)	eindeutige ID des Schlüsselbandes, von welchem der Benutzer die Schlüsselinformationen anfordern möchte
WeekInYearStart (int)	Kalenderwoche des Startzeitpunkts
WeekInYearEnd (int)	Kalenderwoche des Endzeitpunkts
YearStart (int)	Jahr des Startzeitpunkts
YearEnd (int)	Jahr des Endzeitpunkts

Response *ListKeysFromKeybandResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
SharingKeyInfoList (List<SharingKeyInfo>)	angeforderte Liste der Schlüsselinformationen jener Schlüssel des Schlüsselbandes im gewünschten Zeitraum

Zusätzlicher Typ *SharingKeyInfo*

Beschreibung siehe A.6.1

Funktion *AddKeyAccess*

Mit dieser Funktion kann ein Schlüsselbesitzer (Kunde) einem Benutzer (Schlüsselanforderer) den Zugriff auf die Private-Keys der angegebenen Schlüsselpaare gewähren. Somit kann der Schlüsselanforderer nach der Ausführung dieser Funktion alle Datensätze, welche mit dem Public-Key dieser Schlüsselpaare verschlüsselt sind, anfordern und im Klartext lesen.

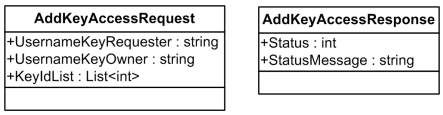


Abbildung A.37: Klassendiagramme der Klassen *AddKeyAccessRequest* und *AddKeyAccessResponse*

Request *AddKeyAccessRequest*

Attribut	Beschreibung
UsernameKeyRequester (string)	Benutzername des Schlüsselanforderers
UsernameKeyOwner (string)	Benutzername des Schlüsselbesitzers
KeyIdList (List<int>)	Liste der Schlüssel-IDs jener Schlüsselpaare, auf welche der Schlüsselbesitzer den Zugriff durch den Schlüsselanforderer gewähren möchte.

Response *AddKeyAccesstResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *ListKeybandShares*

Mit dieser Funktion kann der Schlüsselbesitzer eine Liste von Schlüssel-IDs anfordern, welche für jeden Benutzer, welcher je Zugriff auf Schlüssel eines Schlüsselbands erhalten hat, mindestens einen Schlüssel enthält. Damit kann bei der Inventarisierung eines neuen Zählers garantiert werden, dass jeder Benutzer, welcher Zugriff auf Wochenschlüssel dieser Immobilie (dieses Schlüsselbandes) hat, den neuen Zählerdatensatz ebenfalls lesen kann (sofern der Zählerdatensatz mit all den gelieferten Schlüsseln verschlüsselt wird).

Diese Funktion darf von allen Benutzern aufgerufen werden.

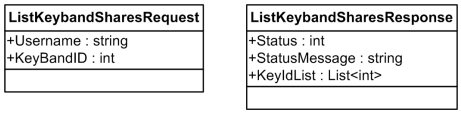


Abbildung A.38: Klassendiagramme der Klassen *ListKeybandSharesRequest* und *ListKeybandSharesResponse*

Request *ListKeybandSharesRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Schlüsselbesitzers
KeyBandId (int)	eindeutige ID des Schlüsselbandes, für welche die Liste angefordert werden soll

Response *ListKeybandSharesResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeyIdList (List<int>)	Liste der Schlüssel-IDs, welche für jeden Benutzer, welcher Zugriff auf Schlüssel des Schlüsselbandes erhalten hat, mindestens einen Schlüssel enthält

A.6.2 Interface zwischen automatischer Datenquelle und Schlüsselverwaltung

Diese Schnittstelle dient dazu, dass sich automatische Datenquellen mit neuen (aktuellen) Wochenschlüsseln versorgen können.

Funktion *GetPublicKeyFromKeybandForAutomaticMeter*

Diese Funktion dient dazu, den Public-Key eines Wochenschlüsselpaares anzufordern. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen danach mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle digital zu signieren.

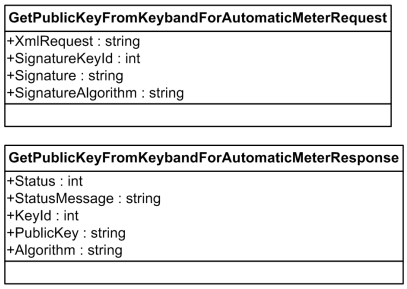


Abbildung A.39: Klassendiagramme der Klassen *GetPublicKeyFromKeybandForAutomaticMeterRequest* und *GetPublicKeyFromKeybandForAutomaticMeterResponse*

Request *GetPublicKeyFromKeybandForAutomaticMeterRequest*

Attribut	Beschreibung
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature (string)	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *GetPublicKeyFromKeybandForAutomaticMeterResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
KeyId (int)	eindeutige ID des Wochenschlüsselpaares
PublicKey (string)	der angeforderte Public-Key im XML-Format
Algorithm (string)	Name des Verschlüsselungsalgorithmus, für welchen der Schlüssel vorgesehen ist. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

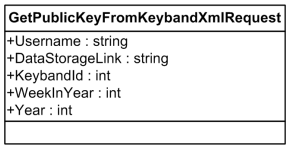


Abbildung A.40: Klassendiagramme des Typs *GetPublicKeyFromKeybandXmlRequest*

Typ des zu serialisierenden Requests (*GetPublicKeybandXmlRequest*)

Attribut	Beschreibung
Username (string)	Benutzername des Schlüsselbesitzers
DataStorageLink (string)	URL des Datenspeichers, in welchen der automatische Zähler, welcher Schlüssel dieses Schlüsselbandes verwendt, Daten speichert.
KeybandId (int)	eindeutige ID des Schlüsselbandes, für welche der Schlüssel angefordert wird
WeekInYear (int)	Kalenderwoche, für welche der Schlüssel angefordert wird.
Year (int)	Jahr, für welches der Schlüssel angefordert wird.

A.6.3 Interface zwischen Datenspeicher und Schlüsselverwaltung

Diese Schnittstelle dient dazu, dass ein Datenspeicher, welcher einen Request von einer automatischen Datenquelle erhält (welcher immer digital signiert ist), den Public-Key des Signaturschlüsselpaares anfordern kann, um im Anschluss die Signatur auf Gültigkeit zu überprüfen.

Funktion *GetPublicKeyForSignature*

Mit dieser Funktion kann der Datenspeicher den Public-Key eines Signaturschlüsselpaares anfordern.

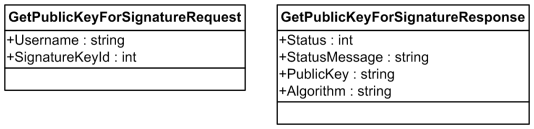


Abbildung A.41: Klassendiagramme der Klassen *GetPublicKeyForSignatureRequest* und *GetPublicKeyForSignatureResponse*

Request *GetPublicKeyForSignatureRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Schlüsselbesitzers.
SignatureKeyId (int)	eindeutige Schlüssel-ID des Signaturschlüsselpaares

Response *GetPublicKeyForSignatureResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
PublicKey (string)	angeforderter Public-Key des Signaturschlüsselpaares im XML-Format
Algorithm (string)	Name des Signaturalgorithmus, für welchen der Schlüssel verwendet wird. Zur Zeit sind folgende Algorithmen vorhanden: *RSA* - RSA-Algorithmus

A.6.4 Interface zwischen Client und Datenspeicher

Funktion *AddNewUser*

Diese Funktion wird dazu verwendet, um einen neuen Benutzer beim Datenspeicher anzulegen. Dabei wird geprüft, ob schon ein Benutzer mit dem übergebenen Benutzernamen existiert. Sollte dies der Fall sein, so wird ein Fehler zurückgegeben. Sollte der Benutzername noch frei sein, so wird ein neuer Benutzer angelegt.

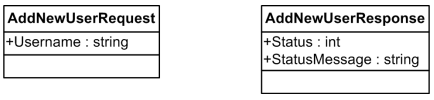


Abbildung A.42: Klassendiagramme der Klassen *AddNewUserRequest* und *AddNewUserResponse*

Request *AddNewUserRequest*

Attribut	Beschreibung
Username (string)	der Benutzername des neu anzulegenden Benutzers

Response *AddNewUserResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *ListData*

Mit dieser Funktion kann ein Benutzer eine Liste der gespeicherten Datensätzes eines Kunden anfordern. Dabei kann der Benutzer die Liste auf einen bestimmten Typ von Datensätzen (Daten, Metadaten, Wurzelknoten der Metadaten) beschränken. Der Benutzer muss eine Liste von Schlüssel-IDs angeben, mit welchen er die Datensätze entschlüsseln kann. Somit kann beispielsweise ein Berater anfragen, welche Datensätze er mit jenen Schlüsseln, welche der Kunde für ihn freigegeben hat, entschlüsseln und somit lesen kann.
Diese Funktion darf von allen Benutzern aufgerufen werden.

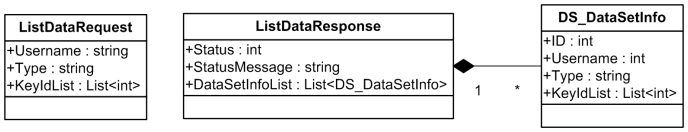


Abbildung A.43: Klassendiagramme der Klassen *ListDataRequest*, *ListDataResponse* und *DS_DataSetInfo*

Request *ListDataRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Kunden
Type (string)	Art des Datensatzes. Dieses Attribut dient dazu, nur Datensätze eines bestimmten Typs zu suchen. Momentan sind folgende Werte verfügbar: *data* - normaler Datensatz *metadata* - Datensatz welcher Metadaten enthält *metadataRoot* - Datensatz, welcher den Wurzelknoten der Metadaten dieses Kunden bildet

Attribut	Beschreibung
KeyIdList (List<int>)	Liste von Schlüssel-IDs jener Schlüssel, mit welchen der Benutzer die Datensätze entschlüsseln möchte

Response *ListDataResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
DataSetInfoList (List<DS_DataSetInfo>)	Liste der Datensatzinformationen

Zusätzlicher Typ *DS_DataSetInfo*

Attribut	Beschreibung
ID (int)	ID des Datensatzes in diesem Datenspeicher
Username (string)	Benutzername des Kunden, welchem die Daten gehören
Type (string)	Art des Datensates. Zur Zeit sind folgende Werte verfügbar: *data* - normaler Datensatz *metadata* - Datensatz, welcher Metadaten enthält *metadataRoot* - Datensatz, welcher den Wurzelknoten der Metadaten dieses Kunden bildet
KeyIdList (List<int>)	Liste von Schlüssel-IDs, mit welchen der Datensatz entschlüsselt werden kann.

Funktion *GetData*

Mithilfe dieser Funktion kann ein Benutzer Datensätze eines Kunden anfordern. Dabei muss der Benutzer für jeden anzufordernden Datensatz die Schlüssel-ID jenes Schlüssels, mit welchem er den Datensatz im Anschluss entschlüsseln möchte, angeben.
Diese Funktion darf von allen Benutzern aufgerufen werden.

Request *GetDataRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Kunden

Attribut	Beschreibung
IdKeyIdList (List<DS_GetDataRequestItem>)	Liste der anzufordernden Datensätze

Zusätzlicher Typ *DS_GetDataRequestItem*

Attribut	Beschreibung
ID (int)	ID des anzufordernden Datensatzes
KeyId (int)	Schlüssel-ID jenes Schlüssels, mit welchem der Benutzer den Datensatz entschlüsseln möchte

Response *GetDataResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
DataSetList (List<DS_DataSet>)	Liste der Datensätze

Zusätzlicher Typ *DS_DataSet*

Attribut	Beschreibung
ID (int)	ID des Datensatzes in diesem Datenspeicher
Username (string)	Benutzername des Kunden, welchem die Daten gehören
Type (string)	Art des Datensates; zur Zeit sind folgende Werte verfügbar: *data* - normaler Datensatz *metadata* - Datensatz welcher Metadaten enthält *metadataRoot* - Datensatz, welcher den Wurzelknoten der Metadaten dieses Kunden bildet
EncryptedData (byte[])	verschlüsselte Nutzdaten des Datensatzes
EncryptionAlgorithm (string)	Verschlüsselungsalgorithmus, mit welchem die Nutzdaten des Datensatzes verschlüsselt wurden; zur Zeit sind folgende Algorithmen verfügbar: *AES* - AES Algorithmus
KeyList (List<DS_KeyInfo>)	Liste der verschlüsselten Session-Keys (inkl. zugehörige Informationen)

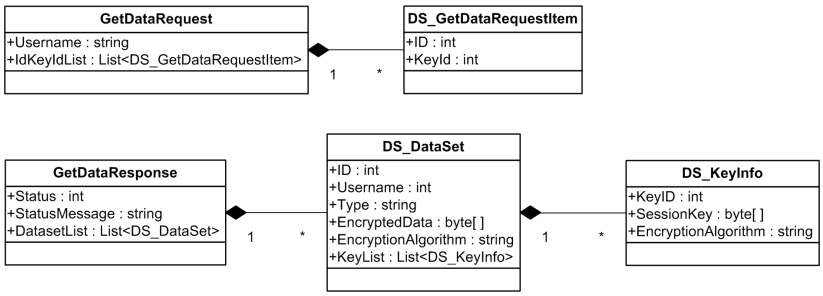


Abbildung A.44: Klassendiagramme der Klassen *GetDataRequest*, *GetDataResponse*, *DS_GetDataRequestItem*, *DS_DataSet* und *DS_KeyInfo*

Zusätzlicher Typ *DS_KeyInfo*

Attribut	**Beschreibung**
KeyID (int)	Schlüssel-ID des Schlüsselpaares, mit welchem der Session-Key verschlüsselt wurde
SessionKey (byte[])	verschlüsselter Session-Key
EncryptionAlgorithm (string)	Verschlüsselungsalgorithmus, mit welchem der Session-Key verschlüsselt wurde; zur Zeit sind folgende Algorithmen verfügbar: *RSA* - RSA Algorithmus

Funktion *GetNewIds*

Mit dieser Funktion kann ein Kunde neue Datensatz-IDs zur Speicherung neuer Datensätze reservieren.
Diese Funktion darf nur vom Kunden selbst ausgeführt werden.

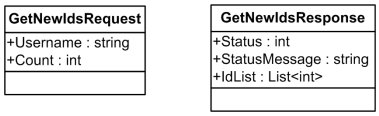


Abbildung A.45: Klassendiagramme der Klassen *GetNewIdsRequest* und *GetNewIdsResponse*

Request *GetNewIdsRequest*

Attribut	**Beschreibung**
Username (string)	der Benutzername des neu anzulegenden Benutzers
Count (int)	Anzahl der zu reservierenden Datensatz-IDs

Response *GetNewIdsResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
IdList (List<int>)	Liste der neuen Datensatz-IDs

Funktion *AddUpdateData*

Mit dieser Funktion kann ein Kunde mehrere Datensätze gleichzeitig aktualisieren oder neu hinzufügen. Damit die Funktion erfolgreich ausgeführt werden kann, muss der Request den Lockstring des aktuell aktivierten Locks des Datenspeichers beinhalten. Sollte dies nicht der Fall sein, gibt die Funktion einen Fehler zurück. Sollte während des Speicherns der Datensätze ein Fehler auftreten, so wird ebenfalls ein Fehler retourniert und alle bis dahin vorgenommenen Veränderungen rückgängig gemacht.
Diese Funktion darf ausschließlich vom Kunden selbst aufgerufen werden.

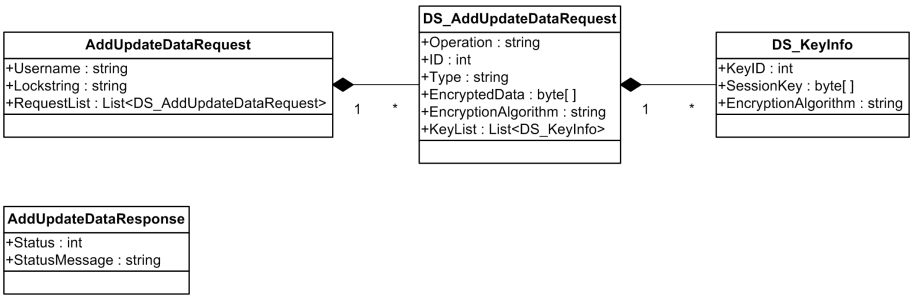


Abbildung A.46: Klassendiagramme der Klassen *AddUpdateDataRequest*, *AddUpdateDataResponse*, *DS_AddUpdateDataRequest* und *DS_KeyInfo*

Request *AddUpdateDataRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Kunden
Lockstring (string)	Lockstring des Kunden für diesen Datenspeicher

Attribut	Beschreibung
RequestList (List<DS_AddUpdateDataRequest>)	Liste der neuen bzw. aktualisierten Datensätze

Zusätzlicher Typ *DS_AddUpdateDataRequest*

Attribut	Beschreibung
Operation (string)	dieses Attribut gibt an, ob es sich um einen neuen Datensatz handelt oder ob der Datensatz aktualisiert werden soll; folgende Operationen sind verfügbar: *add* - hinzufügen eines neuen Datensatzes *update* - aktualisieren eines bestehenden Datensatzes
ID (int)	ID des neuen bzw. zu aktualisierenden Datensatzes
Type (string)	Art des Datensatzes; zur Zeit sind folgende Werte verfügbar: *data* - normaler Datensatz *metadata* - Datensatz welcher Metadaten enthält *metadataRoot* - Datensatz, welcher den Wurzelknoten der Metadaten dieses Kunden bildet
EncryptedData (byte[])	verschlüsselte Nutzdaten des Datensatzes
EncryptionAlgorithm (string)	Verschlüsselungsalgorithmus, mit welchem die Nutzdaten verschlüsselt wurden; zur Zeit sind folgende Algorithmen verfügbar: *AES* - AES Algorithmus
KeyList (List<DS_KeyInfo>)	Liste der verschlüsselten Session-Keys (inkl. zugehörige Informationen)

Zusätzlicher Typ *DS_KeyInfo*

Beschreibung siehe A.6.4

Response *AddUpdateDataResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

Funktion *SubscribeLock*

Diese Funktion dient zum Anfordern des Locks des Kunden beim Datenspeicher. Die Funktion darf nur vom Kunden selbst aufgerufen werden.

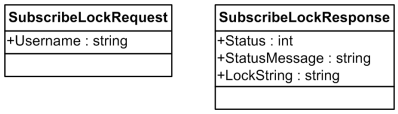


Abbildung A.47: Klassendiagramme der Klassen *SubscribeLockRequest* und *SubscribeLockResponse*

Request *SubscribeLockRequest*

Attribut	Beschreibung
Username (string)	Benutzername des Kunden

Response *SubscribeLockResponse*

Attribut	Beschreibung
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.
Lockstring (string)	Lockstring, welcher das Lock eindeutig identifiziert. Wenn $Lockstring \neq null$, hat der Client das Lock sofort erhalten. $Lockstring = null$ bedeutet, dass der Client auf das Lock warten muss und später, sobald das Lock frei wird, ein Callback erhält.

Callbackfunktion *NotifyLock*

Diese Funktion wird vom Datenspeicher auf der Clientseite aufgerufen. Sie dient dazu, den Client über den Erhalt des Locks zu informieren. Diese Funktion hat keinen Response (One-Way-Funktion). Dieses hat den Vorteil, dass der Serverthread des Datenspeichers nicht auf eine Antwort des Clients warten muss.

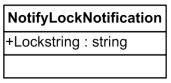


Abbildung A.48: Klassendiagramme der Klassen *NotifyLockNotification*

Request vom Datenspeicher zum Client (*NotifyLockNotification*)

Attribut	Beschreibung
Lockstring (string)	Lockstring, welcher das erhaltene Lock eindeutig identifiziert.

Funktion *ReleaseLock*

Diese Funktion dient zur Freigabe des Locks. Die Funktion darf nur vom Kunden selbst aufgerufen werden.

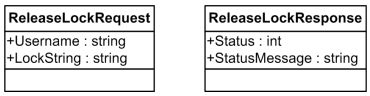


Abbildung A.49: Klassendiagramme der Klassen *ReleaseLockRequest* und *ReleaseLockResponse*

Request *ReleaseLockRequest*

Attribut	**Beschreibung**
Username (string)	Benutzername des Kunden
Lockstring (string)	Lockstring, welcher das Lock, welches freigegeben werden soll, eindeutig identifiziert.

Response *ReleaseLockResponse*

Attribut	**Beschreibung**
Status (int)	Der Status gibt an, ob die Funktion erfolgreich ausgeführt wurde. $Status = 0$ bedeutet erfolgreich ausgeführt. $Status \neq 0$ deutet auf einen Fehler hin. Nähere Informationen zum Fehler bekommt man durch die Auswertung des Attributs *StatusMessage*.
StatusMessage (string)	Dieses Attribut enthält bei Fehlern nähere Informationen zum Fehler bzw. zur Fehlerursache.

A.6.5 Interface zwischen automatischer Datenquelle und Datenspeicher

Funktion *GetDataForAutomaticMeter*

Mit dieser Funktion kann die automatische Datenquelle Datensätze des Kunden vom Datenspeicher anfordern. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen danach mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle digital zu signieren.

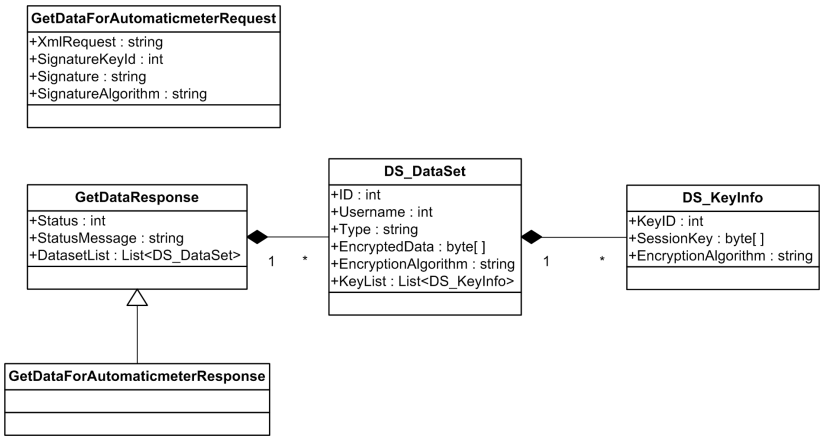


Abbildung A.50: Klassendiagramme der Klassen *GetDataForAutomaticmeterRequest* und *GetDataForAutomaticmeterResponse*

Request *GetDataForAutomaticmeterRequest*

Attribut	**Beschreibung**
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature (string)	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *GetDataForAutomaticmeterResponse*

Dieser Typ erbt direkt vom Typ *GetDataResponse*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

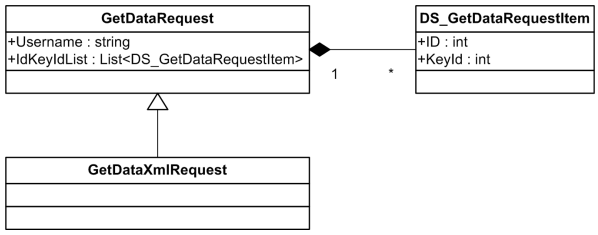


Abbildung A.51: Klassendiagramm des Typs *GetDataXmlRequest*

Typ des zu serialisierenden Requests (*GetDataXmlRequest*)

Dieser Typ erbt direkt vom Typ *GetDataRequest*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

Funktion *GetNewIdsForAutomaticMeter*

Mit dieser Funktion kann die automatische Datenquelle IDs für neue Datensätze beim Datenspeicher anfordern. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen danach mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle, digital zu signieren.

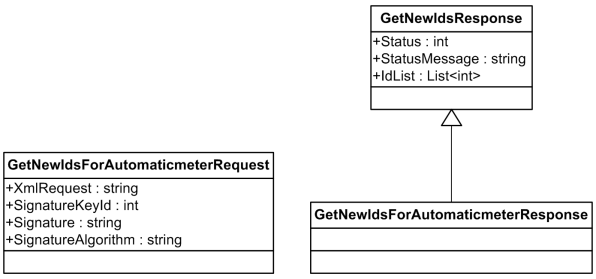


Abbildung A.52: Klassendiagramme der Klassen *GetNewIdsForAutomaticmeterRequest* und *GetNewIdsForAutomaticmeterResponse*

Request *GetNewIdsForAutomaticmeterRequest*

Attribut	**Beschreibung**
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature (string)	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *GetNewIdsForAutomaticmeterResponse*

Dieser Typ erbt direkt vom Typ *GetNewIdsResponse*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

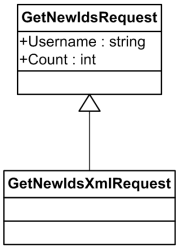


Abbildung A.53: Klassendiagramm des Typs *GetNewIdsXmlRequest*

Typ des zu serialisierenden Requests (*GetNewIdsXmlRequest*)

Dieser Typ erbt direkt vom Typ *GetNewIdsRequest*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

Funktion *AddUpdateDataForAutomaticMeter*

Mit dieser Funktion kann die automatische Datenquelle neue und/oder aktualisierte Datensätze im Datenspeicher sichern. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen anschließend mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle digital zu signieren.

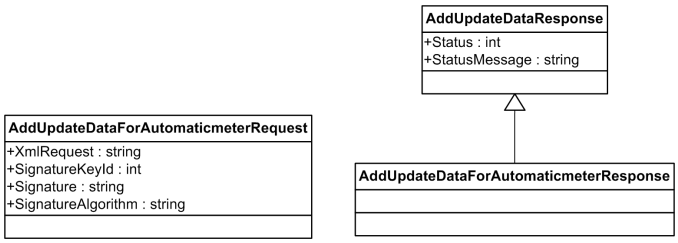


Abbildung A.54: Klassendiagramme der Klassen *AddUpdateDataForAutomaticmeterRequest* und *AddUpdateDataForAutomaticmeterResponse*

Request *AddUpdateDataForAutomaticmeterRequest*

Attribut	Beschreibung
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature (string)	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *AddUpdateDataForAutomaticmeterResponse*

Dieser Typ erbt direkt vom Typ *GetNewIdsResponse*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

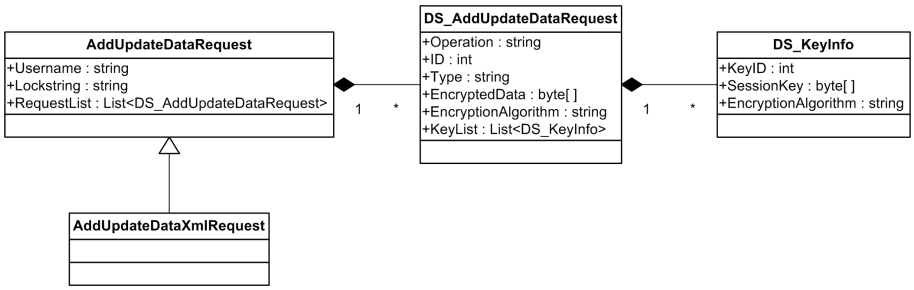


Abbildung A.55: Klassendiagramm des Typs *AddUpdateDataXmlRequest*

Typ des zu serialisierenden Requests (*AddUpdateDataXmlRequest*)

Dieser Typ erbt direkt vom Typ *AddUpdateDataRequest*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

Funktion *SubscribeLockForAutomaticMeter*

Mit dieser Funktion kann die automatische Datenquelle das Lock des Kunden beim Datenspeicher anfordern. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen danach mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle, digital zu signieren.

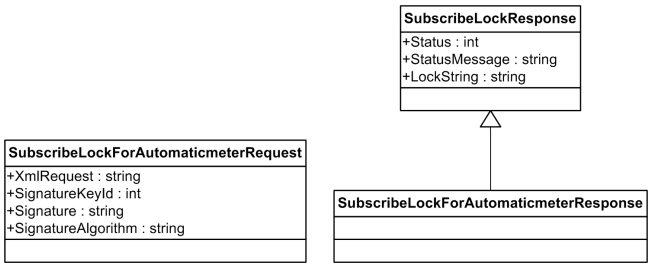


Abbildung A.56: Klassendiagramme der Klassen *SubscribeLockForAutomaticmeterRequest* und *SubscribeLockForAutomaticmeterResponse*

Request *SubscribeLockForAutomaticmeterRequest*

Attribut	**Beschreibung**
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *SubscribeLockForAutomaticmeterResponse*

Dieser Typ erbt direkt vom Typ *SubscribeLockResponse*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

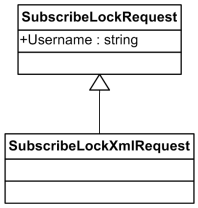


Abbildung A.57: Klassendiagramm des Typs *SubscribeLockXmlRequest*

Typ des zu serialisierenden Requests (*SubscribeLockXmlRequest*)

Dieser Typ erbt direkt vom Typ *SubscribeLockRequest*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

Callbackfunktion *NotifyLock*

Diese Funktion wird vom Datenspeicher auf der Clientseite aufgerufen. Sie dient dazu, den Client über den Erhalt des Locks zu informieren. Diese Funktion hat keinen Response (One-Way-Funktion). Dieses hat den Vorteil, dass der Serverthread des Datenspeichers nicht auf eine Antwort des Clients warten muss.

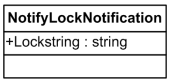


Abbildung A.58: Klassendiagramme der Klassen *NotifyLockNotification*

Request vom Datenspeicher zum Client (*NotifyLockNotification*)

Attribut	Beschreibung
Lockstring (string)	Lockstring, welcher das erhaltene Lock eindeutig identifiziert.

Funktion *ReleaseLockForAutomaticMeter*

Mit dieser Funktion kann die automatische Datenquelle das Lock des Kunden beim Datenspeicher freigeben. Die Nutzdaten des Requests dieser Funktion werden von der automatischen Datenquelle händisch ins XML-Format serialisiert, um diesen danach mit dem parametrierten Private-Key des Signaturschlüsselpaares der Datenquelle, digital zu signieren.

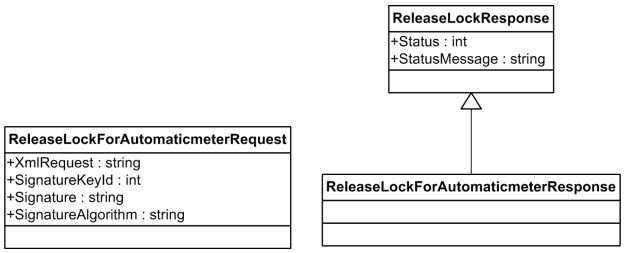


Abbildung A.59: Klassendiagramme der Klassen *ReleaseLockForAutomaticmeterRequest* und *ReleaseLockForAutomaticmeterResponse*

Request *ReleaseLockForAutomaticmeterRequest*

Attribut	Beschreibung
XmlRequest (string)	Nutzdaten des Requests im XML-Format
SignatureKeyId (int)	eindeutige Schlüsselpaar-ID des Signaturschlüsselpaares
Signature (string)	Signatur des Requests (Base64 kodiert)
SignatureAlgorithm (string)	Name des Signaturalgorithmus - Zur Zeit sind folgende Namen zulässig: *RSA-SHA1* - der SHA1-Algorithmus wird zur Berechnung des Hashwertes verwendet; anschließend wird dieser Hashwert mit dem Private-Key des Signaturschlüsselpaares mittels RSA-Algorithmus signiert

Response *ReleaseLockForAutomaticmeterResponse*

Dieser Typ erbt direkt vom Typ *ReleaseLockResponse*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

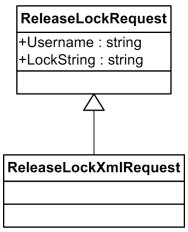


Abbildung A.60: Klassendiagramm des Typs *ReleaseLockXmlRequest*

Typ des zu serialisierenden Requests (*ReleaseLockXmlRequest*)

Dieser Typ erbt direkt vom Typ *ReleaseLockRequest*, ohne diesen zu erweitern. Deshalb weist dieser Typ die selben Attribute wie die Elternklasse auf.

A.7 Listings der WSDL-Files und XML-Schemas der Interfaces

A.7.1 Interface zwischen Client und Schlüsselverwaltung

http://localhost:8010/?wsdl

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.
    xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
    wss-wssecurity-utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:tns="http://tempuri.org
    /" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas
    .xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/
    addressing/policy" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas
    .microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsaw="http://www.w3.org/2006/05/addressing
    /wsdl" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:wsa10="http://www.w3.
    org/2005/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" name="
    KeyManagementService" targetNamespace="http://tempuri.org/">
  <wsp:Policy wsu:Id="wsHttpBinding_policy">
    <wsp:ExactlyOne>
      <wsp:All>
        <sp:SymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
          <wsp:Policy>
            <sp:ProtectionToken>
              <wsp:Policy>
                <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws
                    /2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
                  <wsp:Policy>
                    <sp:RequireDerivedKeys/>
                    <sp:BootstrapPolicy>
                      <wsp:Policy>
                        <sp:SignedParts>
                          <sp:Body/>
                          <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
                              "/>
                          <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                          <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                          <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                          <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                          <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                          <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/
                              addressing"/>
                        </sp:SignedParts>
                        <sp:EncryptedParts>
                          <sp:Body/>
                        </sp:EncryptedParts>
                        <sp:SymmetricBinding>
                          <wsp:Policy>
                            <sp:ProtectionToken>
                              <wsp:Policy>
                                <sp:SpnegoContextToken sp:IncludeToken="http://schemas.xmlsoap
                                    .org/ws/2005/07/securitypolicy/IncludeToken/
                                    AlwaysToRecipient">
                                  <wsp:Policy>
                                    <sp:RequireDerivedKeys/>
                                  </wsp:Policy>
                                </sp:SpnegoContextToken>
                              </wsp:Policy>
                            </sp:ProtectionToken>
                            <sp:AlgorithmSuite>
                              <wsp:Policy>
                                <sp:Basic256/>
                              </wsp:Policy>
                            </sp:AlgorithmSuite>
                            <sp:Layout>
                              <wsp:Policy>
                                <sp:Strict/>
                              </wsp:Policy>
                            </sp:Layout>
                            <sp:IncludeTimestamp/>
                            <sp:EncryptSignature/>
```

```
51                              <sp:OnlySignEntireHeadersAndBody/>
52                            </wsp:Policy>
53                          </sp:SymmetricBinding>
54                          <sp:Wss11>
55                            <wsp:Policy>
56                              <sp:MustSupportRefKeyIdentifier/>
57                              <sp:MustSupportRefIssuerSerial/>
58                              <sp:MustSupportRefThumbprint/>
59                              <sp:MustSupportRefEncryptedKey/>
60                            </wsp:Policy>
61                          </sp:Wss11>
62                          <sp:Trust10>
63                            <wsp:Policy>
64                              <sp:MustSupportIssuedTokens/>
65                              <sp:RequireClientEntropy/>
66                              <sp:RequireServerEntropy/>
67                            </wsp:Policy>
68                          </sp:Trust10>
69                        </wsp:Policy>
70                      </sp:BootstrapPolicy>
71                    </wsp:Policy>
72                  </sp:SecureConversationToken>
73                </wsp:Policy>
74              </sp:ProtectionToken>
75              <sp:AlgorithmSuite>
76                <wsp:Policy>
77                  <sp:Basic256/>
78                </wsp:Policy>
79              </sp:AlgorithmSuite>
80              <sp:Layout>
81                <wsp:Policy>
82                  <sp:Strict/>
83                </wsp:Policy>
84              </sp:Layout>
85              <sp:IncludeTimestamp/>
86              <sp:EncryptSignature/>
87              <sp:OnlySignEntireHeadersAndBody/>
88            </wsp:Policy>
89          </sp:SymmetricBinding>
90          <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
91            <wsp:Policy>
92              <sp:MustSupportRefKeyIdentifier/>
93              <sp:MustSupportRefIssuerSerial/>
94              <sp:MustSupportRefThumbprint/>
95              <sp:MustSupportRefEncryptedKey/>
96            </wsp:Policy>
97          </sp:Wss11>
98          <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
99            <wsp:Policy>
100             <sp:MustSupportIssuedTokens/>
101             <sp:RequireClientEntropy/>
102             <sp:RequireServerEntropy/>
103           </wsp:Policy>
104         </sp:Trust10>
105         <wsaw:UsingAddressing/>
106       </wsp:All>
107     </wsp:ExactlyOne>
108   </wsp:Policy>
109   <wsp:Policy wsu:Id="wsHttpBinding_AddNewUser_Input_policy">
110     <wsp:ExactlyOne>
111       <wsp:All>
112         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
113           <sp:Body/>
114           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
115           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
116           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
117           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
118           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
119           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
120           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
121         </sp:SignedParts>
122         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
123           <sp:Body/>
124         </sp:EncryptedParts>
125       </wsp:All>
126     </wsp:ExactlyOne>
127   </wsp:Policy>
128   <wsp:Policy wsu:Id="wsHttpBinding_AddNewUser_output_policy">
```

```
129     <wsp:ExactlyOne>
130       <wsp:All>
131         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
132           <sp:Body/>
133           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
134           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
135           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
136           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
137           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
138           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
139           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
140         </sp:SignedParts>
141         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
142           <sp:Body/>
143         </sp:EncryptedParts>
144       </wsp:All>
145     </wsp:ExactlyOne>
146   </wsp:Policy>
147   <wsp:Policy wsu:Id="wsHttpBinding_ListDataStorage_Input_policy">
148     <wsp:ExactlyOne>
149       <wsp:All>
150         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
151           <sp:Body/>
152           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
153           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
154           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
155           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
156           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
157           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
158           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
159         </sp:SignedParts>
160         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
161           <sp:Body/>
162         </sp:EncryptedParts>
163       </wsp:All>
164     </wsp:ExactlyOne>
165   </wsp:Policy>
166   <wsp:Policy wsu:Id="wsHttpBinding_ListDataStorage_output_policy">
167     <wsp:ExactlyOne>
168       <wsp:All>
169         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
170           <sp:Body/>
171           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
172           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
173           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
174           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
175           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
176           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
177           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
178         </sp:SignedParts>
179         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
180           <sp:Body/>
181         </sp:EncryptedParts>
182       </wsp:All>
183     </wsp:ExactlyOne>
184   </wsp:Policy>
185   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewKeyband_Input_policy">
186     <wsp:ExactlyOne>
187       <wsp:All>
188         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
189           <sp:Body/>
190           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
191           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
192           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
193           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
194           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
195           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
196           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
197         </sp:SignedParts>
198         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
199           <sp:Body/>
200         </sp:EncryptedParts>
201       </wsp:All>
202     </wsp:ExactlyOne>
203   </wsp:Policy>
204   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewKeyband_output_policy">
205     <wsp:ExactlyOne>
206       <wsp:All>
```

```
207         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
208           <sp:Body/>
209           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
210           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
211           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
212           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
213           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
214           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
215           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
216         </sp:SignedParts>
217         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
218           <sp:Body/>
219         </sp:EncryptedParts>
220       </wsp:All>
221     </wsp:ExactlyOne>
222   </wsp:Policy>
223   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewIndividualKey_Input_policy">
224     <wsp:ExactlyOne>
225       <wsp:All>
226         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
227           <sp:Body/>
228           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
229           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
230           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
231           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
232           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
233           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
234           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
235         </sp:SignedParts>
236         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
237           <sp:Body/>
238         </sp:EncryptedParts>
239       </wsp:All>
240     </wsp:ExactlyOne>
241   </wsp:Policy>
242   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewIndividualKey_output_policy">
243     <wsp:ExactlyOne>
244       <wsp:All>
245         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
246           <sp:Body/>
247           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
248           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
249           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
250           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
251           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
252           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
253           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
254         </sp:SignedParts>
255         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
256           <sp:Body/>
257         </sp:EncryptedParts>
258       </wsp:All>
259     </wsp:ExactlyOne>
260   </wsp:Policy>
261   <wsp:Policy wsu:Id="wsHttpBinding_ListKeysOfUser_Input_policy">
262     <wsp:ExactlyOne>
263       <wsp:All>
264         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
265           <sp:Body/>
266           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
267           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
268           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
269           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
270           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
271           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
272           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
273         </sp:SignedParts>
274         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
275           <sp:Body/>
276         </sp:EncryptedParts>
277       </wsp:All>
278     </wsp:ExactlyOne>
279   </wsp:Policy>
280   <wsp:Policy wsu:Id="wsHttpBinding_ListKeysOfUser_output_policy">
281     <wsp:ExactlyOne>
282       <wsp:All>
283         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
284           <sp:Body/>
```

```
285           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
286           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
287           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
288           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
289           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
290           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
291           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
292         </sp:SignedParts>
293         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
294           <sp:Body/>
295         </sp:EncryptedParts>
296       </wsp:All>
297     </wsp:ExactlyOne>
298   </wsp:Policy>
299   <wsp:Policy wsu:Id="wsHttpBinding_GetPublicKey_Input_policy">
300     <wsp:ExactlyOne>
301       <wsp:All>
302         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
303           <sp:Body/>
304           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
305           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
306           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
307           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
308           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
309           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
310           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
311         </sp:SignedParts>
312         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
313           <sp:Body/>
314         </sp:EncryptedParts>
315       </wsp:All>
316     </wsp:ExactlyOne>
317   </wsp:Policy>
318   <wsp:Policy wsu:Id="wsHttpBinding_GetPublicKey_output_policy">
319     <wsp:ExactlyOne>
320       <wsp:All>
321         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
322           <sp:Body/>
323           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
324           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
325           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
326           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
327           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
328           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
329           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
330         </sp:SignedParts>
331         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
332           <sp:Body/>
333         </sp:EncryptedParts>
334       </wsp:All>
335     </wsp:ExactlyOne>
336   </wsp:Policy>
337   <wsp:Policy wsu:Id="wsHttpBinding_GetPrivateKey_Input_policy">
338     <wsp:ExactlyOne>
339       <wsp:All>
340         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
341           <sp:Body/>
342           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
343           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
344           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
345           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
346           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
347           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
348           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
349         </sp:SignedParts>
350         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
351           <sp:Body/>
352         </sp:EncryptedParts>
353       </wsp:All>
354     </wsp:ExactlyOne>
355   </wsp:Policy>
356   <wsp:Policy wsu:Id="wsHttpBinding_GetPrivateKey_output_policy">
357     <wsp:ExactlyOne>
358       <wsp:All>
359         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
360           <sp:Body/>
361           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
362           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
363           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
364           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
365           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
366           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
367           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
368         </sp:SignedParts>
369         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
370           <sp:Body/>
371         </sp:EncryptedParts>
372       </wsp:All>
373     </wsp:ExactlyOne>
374   </wsp:Policy>
375   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewSignatureKey_Input_policy">
376     <wsp:ExactlyOne>
377       <wsp:All>
378         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
379           <sp:Body/>
380           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
381           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
382           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
383           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
384           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
385           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
386           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
387         </sp:SignedParts>
388         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
389           <sp:Body/>
390         </sp:EncryptedParts>
391       </wsp:All>
392     </wsp:ExactlyOne>
393   </wsp:Policy>
394   <wsp:Policy wsu:Id="wsHttpBinding_CreateNewSignatureKey_output_policy">
395     <wsp:ExactlyOne>
396       <wsp:All>
397         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
398           <sp:Body/>
399           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
400           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
401           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
402           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
403           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
404           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
405           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
406         </sp:SignedParts>
407         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
408           <sp:Body/>
409         </sp:EncryptedParts>
410       </wsp:All>
411     </wsp:ExactlyOne>
412   </wsp:Policy>
413   <wsp:Policy wsu:Id="wsHttpBinding_GetPublicKeyFromKeyband_Input_policy">
414     <wsp:ExactlyOne>
415       <wsp:All>
416         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
417           <sp:Body/>
418           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
419           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
420           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
421           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
422           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
423           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
424           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
425         </sp:SignedParts>
426         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
427           <sp:Body/>
428         </sp:EncryptedParts>
429       </wsp:All>
430     </wsp:ExactlyOne>
431   </wsp:Policy>
432   <wsp:Policy wsu:Id="wsHttpBinding_GetPublicKeyFromKeyband_output_policy">
433     <wsp:ExactlyOne>
434       <wsp:All>
435         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
436           <sp:Body/>
437           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
438           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
439           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
440           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
441           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
442           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
443           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
444         </sp:SignedParts>
445         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
446           <sp:Body/>
447         </sp:EncryptedParts>
448       </wsp:All>
449     </wsp:ExactlyOne>
450   </wsp:Policy>
451   <wsp:Policy wsu:Id="wsHttpBinding_GetShareRequestList_Input_policy">
452     <wsp:ExactlyOne>
453       <wsp:All>
454         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
455           <sp:Body/>
456           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
457           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
458           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
459           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
460           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
461           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
462           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
463         </sp:SignedParts>
464         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
465           <sp:Body/>
466         </sp:EncryptedParts>
467       </wsp:All>
468     </wsp:ExactlyOne>
469   </wsp:Policy>
470   <wsp:Policy wsu:Id="wsHttpBinding_GetShareRequestList_output_policy">
471     <wsp:ExactlyOne>
472       <wsp:All>
473         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
474           <sp:Body/>
475           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
476           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
477           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
478           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
479           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
480           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
481           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
482         </sp:SignedParts>
483         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
484           <sp:Body/>
485         </sp:EncryptedParts>
486       </wsp:All>
487     </wsp:ExactlyOne>
488   </wsp:Policy>
489   <wsp:Policy wsu:Id="wsHttpBinding_AddShareRequest_Input_policy">
490     <wsp:ExactlyOne>
491       <wsp:All>
492         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
493           <sp:Body/>
494           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
495           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
496           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
497           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
498           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
499           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
500           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
501         </sp:SignedParts>
502         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
503           <sp:Body/>
504         </sp:EncryptedParts>
505       </wsp:All>
506     </wsp:ExactlyOne>
507   </wsp:Policy>
508   <wsp:Policy wsu:Id="wsHttpBinding_AddShareRequest_output_policy">
509     <wsp:ExactlyOne>
510       <wsp:All>
511         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
512           <sp:Body/>
513           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
514           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
515           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
516           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
517           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
518           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
519           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
520         </sp:SignedParts>
521         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
522           <sp:Body/>
523         </sp:EncryptedParts>
524       </wsp:All>
525     </wsp:ExactlyOne>
526   </wsp:Policy>
527   <wsp:Policy wsu:Id="wsHttpBinding_ListKeysFromKeyband_Input_policy">
528     <wsp:ExactlyOne>
529       <wsp:All>
530         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
531           <sp:Body/>
532           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
533           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
534           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
535           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
536           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
537           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
538           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
539         </sp:SignedParts>
540         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
541           <sp:Body/>
542         </sp:EncryptedParts>
543       </wsp:All>
544     </wsp:ExactlyOne>
545   </wsp:Policy>
546   <wsp:Policy wsu:Id="wsHttpBinding_ListKeysFromKeyband_output_policy">
547     <wsp:ExactlyOne>
548       <wsp:All>
549         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
550           <sp:Body/>
551           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
552           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
553           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
554           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
555           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
556           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
557           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
558         </sp:SignedParts>
559         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
560           <sp:Body/>
561         </sp:EncryptedParts>
562       </wsp:All>
563     </wsp:ExactlyOne>
564   </wsp:Policy>
565   <wsp:Policy wsu:Id="wsHttpBinding_AddKeyAccess_Input_policy">
566     <wsp:ExactlyOne>
567       <wsp:All>
568         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
569           <sp:Body/>
570           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
571           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
572           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
573           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
574           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
575           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
576           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
577         </sp:SignedParts>
578         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
579           <sp:Body/>
580         </sp:EncryptedParts>
581       </wsp:All>
582     </wsp:ExactlyOne>
583   </wsp:Policy>
584   <wsp:Policy wsu:Id="wsHttpBinding_AddKeyAccess_output_policy">
585     <wsp:ExactlyOne>
586       <wsp:All>
587         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
588           <sp:Body/>
589           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
590           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
591           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
592           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
593           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
594           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
595           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
596         </sp:SignedParts>
```

```
597         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
598           <sp:Body/>
599         </sp:EncryptedParts>
600       </wsp:All>
601     </wsp:ExactlyOne>
602   </wsp:Policy>
603   <wsp:Policy wsu:Id="wsHttpBinding_DeleteShareRequest_Input_policy">
604     <wsp:ExactlyOne>
605       <wsp:All>
606         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
607           <sp:Body/>
608           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
609           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
610           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
611           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
612           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
613           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
614           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
615         </sp:SignedParts>
616         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
617           <sp:Body/>
618         </sp:EncryptedParts>
619       </wsp:All>
620     </wsp:ExactlyOne>
621   </wsp:Policy>
622   <wsp:Policy wsu:Id="wsHttpBinding_DeleteShareRequest_output_policy">
623     <wsp:ExactlyOne>
624       <wsp:All>
625         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
626           <sp:Body/>
627           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
628           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
629           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
630           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
631           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
632           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
633           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
634         </sp:SignedParts>
635         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
636           <sp:Body/>
637         </sp:EncryptedParts>
638       </wsp:All>
639     </wsp:ExactlyOne>
640   </wsp:Policy>
641   <wsp:Policy wsu:Id="wsHttpBinding_ListKeybandShares_Input_policy">
642     <wsp:ExactlyOne>
643       <wsp:All>
644         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
645           <sp:Body/>
646           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
647           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
648           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
649           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
650           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
651           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
652           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
653         </sp:SignedParts>
654         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
655           <sp:Body/>
656         </sp:EncryptedParts>
657       </wsp:All>
658     </wsp:ExactlyOne>
659   </wsp:Policy>
660   <wsp:Policy wsu:Id="wsHttpBinding_ListKeybandShares_output_policy">
661     <wsp:ExactlyOne>
662       <wsp:All>
663         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
664           <sp:Body/>
665           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
666           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
667           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
668           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
669           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
670           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
671           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
672         </sp:SignedParts>
673         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
674           <sp:Body/>
```

```
675         </sp:EncryptedParts>
676       </wsp:All>
677     </wsp:ExactlyOne>
678   </wsp:Policy>
679   <wsdl:types>
680     <xsd:schema targetNamespace="http://tempuri.org/Imports">
681       <xsd:import schemaLocation="http://localhost:8010/?xsd=xsd0" namespace="http://tempuri.
              org/"/>
682       <xsd:import schemaLocation="http://localhost:8010/?xsd=xsd1" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/"/>
683       <xsd:import schemaLocation="http://localhost:8010/?xsd=xsd2" namespace="http://schemas.
              datacontract.org/2004/07/KeyManagement"/>
684       <xsd:import schemaLocation="http://localhost:8010/?xsd=xsd3" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/Arrays"/>
685     </xsd:schema>
686   </wsdl:types>
687   <wsdl:message name="AddNewUserReqeust">
688     <wsdl:part name="parameters" element="tns:AddNewUserReqeust"/>
689   </wsdl:message>
690   <wsdl:message name="AddNewUserResponse">
691     <wsdl:part name="parameters" element="tns:AddNewUserResponse"/>
692   </wsdl:message>
693   <wsdl:message name="ListDataStorageRequest">
694     <wsdl:part name="parameters" element="tns:ListDataStorageRequest"/>
695   </wsdl:message>
696   <wsdl:message name="ListDataStorageResponse">
697     <wsdl:part name="parameters" element="tns:ListDataStorageResponse"/>
698   </wsdl:message>
699   <wsdl:message name="CreateNewKeybandRequest">
700     <wsdl:part name="parameters" element="tns:CreateNewKeybandRequest"/>
701   </wsdl:message>
702   <wsdl:message name="CreateNewKeybandResponse">
703     <wsdl:part name="parameters" element="tns:CreateNewKeybandResponse"/>
704   </wsdl:message>
705   <wsdl:message name="CreateNewIndividualKeyRequest">
706     <wsdl:part name="parameters" element="tns:CreateNewIndividualKeyRequest"/>
707   </wsdl:message>
708   <wsdl:message name="CreateNewIndividualKeyResponse">
709     <wsdl:part name="parameters" element="tns:CreateNewIndividualKeyResponse"/>
710   </wsdl:message>
711   <wsdl:message name="ListKeysOfUserRequest">
712     <wsdl:part name="parameters" element="tns:ListKeysOfUserRequest"/>
713   </wsdl:message>
714   <wsdl:message name="ListKeysOfUserResponse">
715     <wsdl:part name="parameters" element="tns:ListKeysOfUserResponse"/>
716   </wsdl:message>
717   <wsdl:message name="GetPublicKeyRequest">
718     <wsdl:part name="parameters" element="tns:GetPublicKeyRequest"/>
719   </wsdl:message>
720   <wsdl:message name="GetPublicKeyResponse">
721     <wsdl:part name="parameters" element="tns:GetPublicKeyResponse"/>
722   </wsdl:message>
723   <wsdl:message name="GetPrivateKeyRequest">
724     <wsdl:part name="parameters" element="tns:GetPrivateKeyRequest"/>
725   </wsdl:message>
726   <wsdl:message name="GetPrivateKeyResponse">
727     <wsdl:part name="parameters" element="tns:GetPrivateKeyResponse"/>
728   </wsdl:message>
729   <wsdl:message name="CreateNewSignatureKeyRequest">
730     <wsdl:part name="parameters" element="tns:CreateNewSignatureKeyRequest"/>
731   </wsdl:message>
732   <wsdl:message name="CreateNewSignatureKeyResponse">
733     <wsdl:part name="parameters" element="tns:CreateNewSignatureKeyResponse"/>
734   </wsdl:message>
735   <wsdl:message name="GetPublicKeyFromKeybandRequest">
736     <wsdl:part name="parameters" element="tns:GetPublicKeyFromKeybandRequest"/>
737   </wsdl:message>
738   <wsdl:message name="GetPublicKeyFromKeybandResponse">
739     <wsdl:part name="parameters" element="tns:GetPublicKeyFromKeybandResponse"/>
740   </wsdl:message>
741   <wsdl:message name="GetShareRequestListRequest">
742     <wsdl:part name="parameters" element="tns:GetShareRequestListRequest"/>
743   </wsdl:message>
744   <wsdl:message name="GetShareRequestListResponse">
745     <wsdl:part name="parameters" element="tns:GetShareRequestListResponse"/>
746   </wsdl:message>
747   <wsdl:message name="AddShareRequestRequest">
748     <wsdl:part name="parameters" element="tns:AddShareRequestRequest"/>
```

```
749   </wsdl:message>
750   <wsdl:message name="AddShareRequestResponse">
751     <wsdl:part name="parameters" element="tns:AddShareRequestResponse"/>
752   </wsdl:message>
753   <wsdl:message name="ListKeysFromKeyBandRequest">
754     <wsdl:part name="parameters" element="tns:ListKeysFromKeyBandRequest"/>
755   </wsdl:message>
756   <wsdl:message name="ListKeysFromKeyBandResponse">
757     <wsdl:part name="parameters" element="tns:ListKeysFromKeyBandResponse"/>
758   </wsdl:message>
759   <wsdl:message name="AddKeyAccessRequest">
760     <wsdl:part name="parameters" element="tns:AddKeyAccessRequest"/>
761   </wsdl:message>
762   <wsdl:message name="AddKeyAccessResponse">
763     <wsdl:part name="parameters" element="tns:AddKeyAccessResponse"/>
764   </wsdl:message>
765   <wsdl:message name="DeleteShareRequestRequest">
766     <wsdl:part name="parameters" element="tns:DeleteShareRequestRequest"/>
767   </wsdl:message>
768   <wsdl:message name="DeleteShareRequestResponse">
769     <wsdl:part name="parameters" element="tns:DeleteShareRequestResponse"/>
770   </wsdl:message>
771   <wsdl:message name="ListKeybandSharesRequest">
772     <wsdl:part name="parameters" element="tns:ListKeybandSharesRequest"/>
773   </wsdl:message>
774   <wsdl:message name="ListKeybandSharesResponse">
775     <wsdl:part name="parameters" element="tns:ListKeybandSharesResponse"/>
776   </wsdl:message>
777   <wsdl:portType name="IKeyManagementService">
778     <wsdl:operation name="AddNewUser">
779       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/AddNewUser" name="
            AddNewUserReqeust" message="tns:AddNewUserReqeust"/>
780       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/AddNewUserResponse"
            name="AddNewUserResponse" message="tns:AddNewUserResponse"/>
781     </wsdl:operation>
782     <wsdl:operation name="ListDataStorage">
783       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/ListDataStorage" name=
            "ListDataStorageRequest" message="tns:ListDataStorageRequest"/>
784       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
            ListDataStorageResponse" name="ListDataStorageResponse" message="
            tns:ListDataStorageResponse"/>
785     </wsdl:operation>
786     <wsdl:operation name="CreateNewKeyband">
787       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/CreateNewKeyband" name
            ="CreateNewKeybandRequest" message="tns:CreateNewKeybandRequest"/>
788       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
            CreateNewKeybandResponse" name="CreateNewKeybandResponse" message="
            tns:CreateNewKeybandResponse"/>
789     </wsdl:operation>
790     <wsdl:operation name="CreateNewIndividualKey">
791       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/CreateNewIndividualKey
            " name="CreateNewIndividualKeyRequest" message="tns:CreateNewIndividualKeyRequest"/
            >
792       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
            CreateNewIndividualKeyResponse" name="CreateNewIndividualKeyResponse" message="
            tns:CreateNewIndividualKeyResponse"/>
793     </wsdl:operation>
794     <wsdl:operation name="ListKeysOfUser">
795       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/ListKeysOfUser" name="
            ListKeysOfUserRequest" message="tns:ListKeysOfUserRequest"/>
796       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
            ListKeysOfUserResponse" name="ListKeysOfUserResponse" message="
            tns:ListKeysOfUserResponse"/>
797     </wsdl:operation>
798     <wsdl:operation name="GetPublicKey">
799       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/GetPublicKey" name="
            GetPublicKeyRequest" message="tns:GetPublicKeyRequest"/>
800       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/GetPublicKeyResponse"
             name="GetPublicKeyResponse" message="tns:GetPublicKeyResponse"/>
801     </wsdl:operation>
802     <wsdl:operation name="GetPrivateKey">
803       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/GetPrivateKey" name="
            GetPrivateKeyRequest" message="tns:GetPrivateKeyRequest"/>
804       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/GetPrivateKeyResponse
            " name="GetPrivateKeyResponse" message="tns:GetPrivateKeyResponse"/>
805     </wsdl:operation>
806     <wsdl:operation name="CreateNewSignatureKey">
```

```
807       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/CreateNewSignatureKey"
                name="CreateNewSignatureKeyRequest" message="tns:CreateNewSignatureKeyRequest"/>
808       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                CreateNewSignatureKeyResponse" name="CreateNewSignatureKeyResponse" message="
                tns:CreateNewSignatureKeyResponse"/>
809     </wsdl:operation>
810     <wsdl:operation name="GetPublicKeyFromKeyband">
811       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/
                GetPublicKeyFromKeyband" name="GetPublicKeyFromKeybandRequest" message="
                tns:GetPublicKeyFromKeybandRequest"/>
812       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                GetPublicKeyFromKeybandResponse" name="GetPublicKeyFromKeybandResponse" message="
                tns:GetPublicKeyFromKeybandResponse"/>
813     </wsdl:operation>
814     <wsdl:operation name="GetShareRequestList">
815       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/GetShareRequestList"
                name="GetShareRequestListRequest" message="tns:GetShareRequestListRequest"/>
816       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                GetShareRequestListResponse" name="GetShareRequestListResponse" message="
                tns:GetShareRequestListResponse"/>
817     </wsdl:operation>
818     <wsdl:operation name="AddShareRequest">
819       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/AddShareRequest" name=
                "AddShareRequestRequest" message="tns:AddShareRequestRequest"/>
820       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                AddShareRequestResponse" name="AddShareRequestResponse" message="
                tns:AddShareRequestResponse"/>
821     </wsdl:operation>
822     <wsdl:operation name="ListKeysFromKeyband">
823       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/ListKeysFromKeyband"
                name="ListKeysFromKeyBandRequest" message="tns:ListKeysFromKeyBandRequest"/>
824       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                ListKeysFromKeybandResponse" name="ListKeysFromKeyBandResponse" message="
                tns:ListKeysFromKeyBandResponse"/>
825     </wsdl:operation>
826     <wsdl:operation name="AddKeyAccess">
827       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/AddKeyAccess" name="
                AddKeyAccessRequest" message="tns:AddKeyAccessRequest"/>
828       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/AddKeyAccessResponse"
                name="AddKeyAccessResponse" message="tns:AddKeyAccessResponse"/>
829     </wsdl:operation>
830     <wsdl:operation name="DeleteShareRequest">
831       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/DeleteShareRequest"
                name="DeleteShareRequestRequest" message="tns:DeleteShareRequestRequest"/>
832       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                DeleteShareRequestResponse" name="DeleteShareRequestResponse" message="
                tns:DeleteShareRequestResponse"/>
833     </wsdl:operation>
834     <wsdl:operation name="ListKeybandShares">
835       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementService/ListKeybandShares"
                name="ListKeybandSharesRequest" message="tns:ListKeybandSharesRequest"/>
836       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementService/
                ListKeybandSharesResponse" name="ListKeybandSharesResponse" message="
                tns:ListKeybandSharesResponse"/>
837     </wsdl:operation>
838   </wsdl:portType>
839   <wsdl:binding name="wsHttpBinding" type="tns:IKeyManagementService">
840     <wsp:PolicyReference URI="#wsHttpBinding_policy"/>
841     <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
842     <wsdl:operation name="AddNewUser">
843       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/AddNewUser" style
                ="document"/>
844       <wsdl:input name="AddNewUserReqeust">
845         <wsp:PolicyReference URI="#wsHttpBinding_AddNewUser_Input_policy"/>
846         <soap12:body use="literal"/>
847       </wsdl:input>
848       <wsdl:output name="AddNewUserResponse">
849         <wsp:PolicyReference URI="#wsHttpBinding_AddNewUser_output_policy"/>
850         <soap12:body use="literal"/>
851       </wsdl:output>
852     </wsdl:operation>
853     <wsdl:operation name="ListDataStorage">
854       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/ListDataStorage"
                style="document"/>
855       <wsdl:input name="ListDataStorageRequest">
856         <wsp:PolicyReference URI="#wsHttpBinding_ListDataStorage_Input_policy"/>
857         <soap12:body use="literal"/>
858       </wsdl:input>
```

```
859       <wsdl:output name="ListDataStorageResponse">
860         <wsp:PolicyReference URI="#wsHttpBinding_ListDataStorage_output_policy"/>
861         <soap12:body use="literal"/>
862       </wsdl:output>
863     </wsdl:operation>
864     <wsdl:operation name="CreateNewKeyband">
865       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/CreateNewKeyband"
              style="document"/>
866       <wsdl:input name="CreateNewKeybandRequest">
867         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewKeyband_Input_policy"/>
868         <soap12:body use="literal"/>
869       </wsdl:input>
870       <wsdl:output name="CreateNewKeybandResponse">
871         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewKeyband_output_policy"/>
872         <soap12:body use="literal"/>
873       </wsdl:output>
874     </wsdl:operation>
875     <wsdl:operation name="CreateNewIndividualKey">
876       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
            CreateNewIndividualKey" style="document"/>
877       <wsdl:input name="CreateNewIndividualKeyRequest">
878         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewIndividualKey_Input_policy"/>
879         <soap12:body use="literal"/>
880       </wsdl:input>
881       <wsdl:output name="CreateNewIndividualKeyResponse">
882         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewIndividualKey_output_policy"/>
883         <soap12:body use="literal"/>
884       </wsdl:output>
885     </wsdl:operation>
886     <wsdl:operation name="ListKeysOfUser">
887       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/ListKeysOfUser"
            style="document"/>
888       <wsdl:input name="ListKeysOfUserRequest">
889         <wsp:PolicyReference URI="#wsHttpBinding_ListKeysOfUser_Input_policy"/>
890         <soap12:body use="literal"/>
891       </wsdl:input>
892       <wsdl:output name="ListKeysOfUserResponse">
893         <wsp:PolicyReference URI="#wsHttpBinding_ListKeysOfUser_output_policy"/>
894         <soap12:body use="literal"/>
895       </wsdl:output>
896     </wsdl:operation>
897     <wsdl:operation name="GetPublicKey">
898       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/GetPublicKey"
            style="document"/>
899       <wsdl:input name="GetPublicKeyRequest">
900         <wsp:PolicyReference URI="#wsHttpBinding_GetPublicKey_Input_policy"/>
901         <soap12:body use="literal"/>
902       </wsdl:input>
903       <wsdl:output name="GetPublicKeyResponse">
904         <wsp:PolicyReference URI="#wsHttpBinding_GetPublicKey_output_policy"/>
905         <soap12:body use="literal"/>
906       </wsdl:output>
907     </wsdl:operation>
908     <wsdl:operation name="GetPrivateKey">
909       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/GetPrivateKey"
            style="document"/>
910       <wsdl:input name="GetPrivateKeyRequest">
911         <wsp:PolicyReference URI="#wsHttpBinding_GetPrivateKey_Input_policy"/>
912         <soap12:body use="literal"/>
913       </wsdl:input>
914       <wsdl:output name="GetPrivateKeyResponse">
915         <wsp:PolicyReference URI="#wsHttpBinding_GetPrivateKey_output_policy"/>
916         <soap12:body use="literal"/>
917       </wsdl:output>
918     </wsdl:operation>
919     <wsdl:operation name="CreateNewSignatureKey">
920       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
            CreateNewSignatureKey" style="document"/>
921       <wsdl:input name="CreateNewSignatureKeyRequest">
922         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewSignatureKey_Input_policy"/>
923         <soap12:body use="literal"/>
924       </wsdl:input>
925       <wsdl:output name="CreateNewSignatureKeyResponse">
926         <wsp:PolicyReference URI="#wsHttpBinding_CreateNewSignatureKey_output_policy"/>
927         <soap12:body use="literal"/>
928       </wsdl:output>
929     </wsdl:operation>
930     <wsdl:operation name="GetPublicKeyFromKeyband">
```

```
931       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
              GetPublicKeyFromKeyband" style="document"/>
932       <wsdl:input name="GetPublicKeyFromKeybandRequest">
933         <wsp:PolicyReference URI="#wsHttpBinding_GetPublicKeyFromKeyband_Input_policy"/>
934         <soap12:body use="literal"/>
935       </wsdl:input>
936       <wsdl:output name="GetPublicKeyFromKeybandResponse">
937         <wsp:PolicyReference URI="#wsHttpBinding_GetPublicKeyFromKeyband_output_policy"/>
938         <soap12:body use="literal"/>
939       </wsdl:output>
940     </wsdl:operation>
941     <wsdl:operation name="GetShareRequestList">
942       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
              GetShareRequestList" style="document"/>
943       <wsdl:input name="GetShareRequestListRequest">
944         <wsp:PolicyReference URI="#wsHttpBinding_GetShareRequestList_Input_policy"/>
945         <soap12:body use="literal"/>
946       </wsdl:input>
947       <wsdl:output name="GetShareRequestListResponse">
948         <wsp:PolicyReference URI="#wsHttpBinding_GetShareRequestList_output_policy"/>
949         <soap12:body use="literal"/>
950       </wsdl:output>
951     </wsdl:operation>
952     <wsdl:operation name="AddShareRequest">
953       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/AddShareRequest"
              style="document"/>
954       <wsdl:input name="AddShareRequestRequest">
955         <wsp:PolicyReference URI="#wsHttpBinding_AddShareRequest_Input_policy"/>
956         <soap12:body use="literal"/>
957       </wsdl:input>
958       <wsdl:output name="AddShareRequestResponse">
959         <wsp:PolicyReference URI="#wsHttpBinding_AddShareRequest_output_policy"/>
960         <soap12:body use="literal"/>
961       </wsdl:output>
962     </wsdl:operation>
963     <wsdl:operation name="ListKeysFromKeyband">
964       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
              ListKeysFromKeyband" style="document"/>
965       <wsdl:input name="ListKeysFromKeyBandRequest">
966         <wsp:PolicyReference URI="#wsHttpBinding_ListKeysFromKeyband_Input_policy"/>
967         <soap12:body use="literal"/>
968       </wsdl:input>
969       <wsdl:output name="ListKeysFromKeyBandResponse">
970         <wsp:PolicyReference URI="#wsHttpBinding_ListKeysFromKeyband_output_policy"/>
971         <soap12:body use="literal"/>
972       </wsdl:output>
973     </wsdl:operation>
974     <wsdl:operation name="AddKeyAccess">
975       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/AddKeyAccess"
              style="document"/>
976       <wsdl:input name="AddKeyAccessRequest">
977         <wsp:PolicyReference URI="#wsHttpBinding_AddKeyAccess_Input_policy"/>
978         <soap12:body use="literal"/>
979       </wsdl:input>
980       <wsdl:output name="AddKeyAccessResponse">
981         <wsp:PolicyReference URI="#wsHttpBinding_AddKeyAccess_output_policy"/>
982         <soap12:body use="literal"/>
983       </wsdl:output>
984     </wsdl:operation>
985     <wsdl:operation name="DeleteShareRequest">
986       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/
              DeleteShareRequest" style="document"/>
987       <wsdl:input name="DeleteShareRequestRequest">
988         <wsp:PolicyReference URI="#wsHttpBinding_DeleteShareRequest_Input_policy"/>
989         <soap12:body use="literal"/>
990       </wsdl:input>
991       <wsdl:output name="DeleteShareRequestResponse">
992         <wsp:PolicyReference URI="#wsHttpBinding_DeleteShareRequest_output_policy"/>
993         <soap12:body use="literal"/>
994       </wsdl:output>
995     </wsdl:operation>
996     <wsdl:operation name="ListKeybandShares">
997       <soap12:operation soapAction="http://tempuri.org/IKeyManagementService/ListKeybandShares
              " style="document"/>
998       <wsdl:input name="ListKeybandSharesRequest">
999         <wsp:PolicyReference URI="#wsHttpBinding_ListKeybandShares_Input_policy"/>
1000        <soap12:body use="literal"/>
1001      </wsdl:input>
```

```
1002        <wsdl:output name="ListKeybandSharesResponse">
1003          <wsp:PolicyReference URI="#wsHttpBinding_ListKeybandShares_output_policy"/>
1004          <soap12:body use="literal"/>
1005        </wsdl:output>
1006      </wsdl:operation>
1007    </wsdl:binding>
1008    <wsdl:service name="KeyManagementService">
1009      <wsdl:port name="wsHttpBinding" binding="tns:wsHttpBinding">
1010        <soap12:address location="http://localhost:8010/KeyManagement"/>
1011        <wsa10:EndpointReference>
1012          <wsa10:Address>http://localhost:8010/KeyManagement</wsa10:Address>
1013          <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
1014            <Upn>lucknerw@pdts.at</Upn>
1015          </Identity>
1016        </wsa10:EndpointReference>
1017      </wsdl:port>
1018    </wsdl:service>
1019  </wsdl:definitions>
```

http://localhost:8010/?xsd=xsd0

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
        elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3     <xs:import schemaLocation="http://localhost:8010/?xsd=xsd2" namespace="http://schemas.
          datacontract.org/2004/07/KeyManagement"/>
4     <xs:import schemaLocation="http://localhost:8010/?xsd=xsd3" namespace="http://schemas.
          microsoft.com/2003/10/Serialization/Arrays"/>
5     <xs:element name="AddNewUserReqeust">
6       <xs:complexType>
7         <xs:sequence>
8           <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
9         </xs:sequence>
10      </xs:complexType>
11    </xs:element>
12    <xs:element name="AddNewUserResponse">
13      <xs:complexType>
14        <xs:sequence>
15          <xs:element name="Status" minOccurs="0" type="xs:int"/>
16          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
17        </xs:sequence>
18      </xs:complexType>
19    </xs:element>
20    <xs:element name="ListDataStorageRequest">
21      <xs:complexType>
22        <xs:sequence/>
23      </xs:complexType>
24    </xs:element>
25    <xs:element name="ListDataStorageResponse">
26      <xs:complexType>
27        <xs:sequence>
28          <xs:element xmlns:q1="http://schemas.datacontract.org/2004/07/KeyManagement" name="
                DataStorages" minOccurs="0" nillable="true" type="q1:ArrayOfdataStorage"/>
29          <xs:element name="Status" minOccurs="0" type="xs:int"/>
30          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
31        </xs:sequence>
32      </xs:complexType>
33    </xs:element>
34    <xs:element name="CreateNewKeybandRequest">
35      <xs:complexType>
36        <xs:sequence>
37          <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
38          <xs:element name="DataStorageLink" minOccurs="0" nillable="true" type="xs:string"/>
39          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
40        </xs:sequence>
41      </xs:complexType>
42    </xs:element>
43    <xs:element name="CreateNewKeybandResponse">
44      <xs:complexType>
45        <xs:sequence>
46          <xs:element name="NewKeybandId" minOccurs="0" type="xs:int"/>
47          <xs:element name="Status" minOccurs="0" type="xs:int"/>
48          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
49        </xs:sequence>
50      </xs:complexType>
51    </xs:element>
52    <xs:element name="CreateNewIndividualKeyRequest">
53      <xs:complexType>
```

```
54      <xs:sequence>
55        <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
56        <xs:element name="DataStorageLink" minOccurs="0" nillable="true" type="xs:string"/>
57        <xs:element name="Purpose" minOccurs="0" nillable="true" type="xs:string"/>
58        <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
59      </xs:sequence>
60    </xs:complexType>
61  </xs:element>
62  <xs:element name="CreateNewIndividualKeyResponse">
63    <xs:complexType>
64      <xs:sequence>
65        <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
66        <xs:element name="PublicKey" minOccurs="0" nillable="true" type="xs:string"/>
67        <xs:element name="Status" minOccurs="0" type="xs:int"/>
68        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
69      </xs:sequence>
70    </xs:complexType>
71  </xs:element>
72  <xs:element name="ListKeysOfUserRequest">
73    <xs:complexType>
74      <xs:sequence>
75        <xs:element name="Purpose" minOccurs="0" nillable="true" type="xs:string"/>
76        <xs:element name="User" minOccurs="0" nillable="true" type="xs:string"/>
77        <xs:element name="UserRequester" minOccurs="0" nillable="true" type="xs:string"/>
78      </xs:sequence>
79    </xs:complexType>
80  </xs:element>
81  <xs:element name="ListKeysOfUserResponse">
82    <xs:complexType>
83      <xs:sequence>
84        <xs:element xmlns:q2="http://schemas.datacontract.org/2004/07/KeyManagement" name="
              Keys" minOccurs="0" nillable="true" type="q2:ArrayOfKeyInfo"/>
85        <xs:element name="Status" minOccurs="0" type="xs:int"/>
86        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
87      </xs:sequence>
88    </xs:complexType>
89  </xs:element>
90  <xs:element name="GetPublicKeyRequest">
91    <xs:complexType>
92      <xs:sequence>
93        <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
94        <xs:element name="User" minOccurs="0" nillable="true" type="xs:string"/>
95      </xs:sequence>
96    </xs:complexType>
97  </xs:element>
98  <xs:element name="GetPublicKeyResponse">
99    <xs:complexType>
100     <xs:sequence>
101       <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
102       <xs:element name="PublicKey" minOccurs="0" nillable="true" type="xs:string"/>
103       <xs:element name="Status" minOccurs="0" type="xs:int"/>
104       <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
105     </xs:sequence>
106   </xs:complexType>
107 </xs:element>
108 <xs:element name="GetPrivateKeyRequest">
109   <xs:complexType>
110     <xs:sequence>
111       <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
112       <xs:element name="UserRequester" minOccurs="0" nillable="true" type="xs:string"/>
113     </xs:sequence>
114   </xs:complexType>
115 </xs:element>
116 <xs:element name="GetPrivateKeyResponse">
117   <xs:complexType>
118     <xs:sequence>
119       <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
120       <xs:element name="PrivateKey" minOccurs="0" nillable="true" type="xs:string"/>
121       <xs:element name="Status" minOccurs="0" type="xs:int"/>
122       <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
123     </xs:sequence>
124   </xs:complexType>
125 </xs:element>
126 <xs:element name="CreateNewSignatureKeyRequest">
127   <xs:complexType>
128     <xs:sequence>
129       <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
130       <xs:element name="DataStorageLink" minOccurs="0" nillable="true" type="xs:string"/>
```

```
131         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
132       </xs:sequence>
133     </xs:complexType>
134   </xs:element>
135   <xs:element name="CreateNewSignatureKeyResponse">
136     <xs:complexType>
137       <xs:sequence>
138         <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
139         <xs:element name="PrivateKey" minOccurs="0" nillable="true" type="xs:string"/>
140         <xs:element name="Status" minOccurs="0" type="xs:int"/>
141         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
142       </xs:sequence>
143     </xs:complexType>
144   </xs:element>
145   <xs:element name="GetPublicKeyFromKeybandRequest">
146     <xs:complexType>
147       <xs:sequence>
148         <xs:element name="DataStorageLink" minOccurs="0" nillable="true" type="xs:string"/>
149         <xs:element name="KeybandId" minOccurs="0" type="xs:int"/>
150         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
151         <xs:element name="WeekInYear" minOccurs="0" type="xs:int"/>
152         <xs:element name="Year" minOccurs="0" type="xs:int"/>
153       </xs:sequence>
154     </xs:complexType>
155   </xs:element>
156   <xs:element name="GetPublicKeyFromKeybandResponse">
157     <xs:complexType>
158       <xs:sequence>
159         <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
160         <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
161         <xs:element name="PublicKey" minOccurs="0" nillable="true" type="xs:string"/>
162         <xs:element name="Status" minOccurs="0" type="xs:int"/>
163         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
164       </xs:sequence>
165     </xs:complexType>
166   </xs:element>
167   <xs:element name="GetShareRequestListRequest">
168     <xs:complexType>
169       <xs:sequence>
170         <xs:element name="UsernameKeyOwner" minOccurs="0" nillable="true" type="xs:string"/>
171         <xs:element name="UsernameKeyRequester" minOccurs="0" nillable="true" type="xs:string"
              />
172       </xs:sequence>
173     </xs:complexType>
174   </xs:element>
175   <xs:element name="GetShareRequestListResponse">
176     <xs:complexType>
177       <xs:sequence>
178         <xs:element xmlns:q3="http://schemas.datacontract.org/2004/07/KeyManagement" name="
              SharingKeyInfoList" minOccurs="0" nillable="true" type="q3:ArrayOfSharingKeyInfo"
              />
179         <xs:element name="Status" minOccurs="0" type="xs:int"/>
180         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
181       </xs:sequence>
182     </xs:complexType>
183   </xs:element>
184   <xs:element name="AddShareRequestRequest">
185     <xs:complexType>
186       <xs:sequence>
187         <xs:element xmlns:q4="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
              "KeyIdList" minOccurs="0" nillable="true" type="q4:ArrayOfint"/>
188         <xs:element name="UsernameKeyOwner" minOccurs="0" nillable="true" type="xs:string"/>
189         <xs:element name="UsernameKeyRequester" minOccurs="0" nillable="true" type="xs:string"
              />
190       </xs:sequence>
191     </xs:complexType>
192   </xs:element>
193   <xs:element name="AddShareRequestResponse">
194     <xs:complexType>
195       <xs:sequence>
196         <xs:element name="Status" minOccurs="0" type="xs:int"/>
197         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
198       </xs:sequence>
199     </xs:complexType>
200   </xs:element>
201   <xs:element name="ListKeysFromKeyBandRequest">
202     <xs:complexType>
203       <xs:sequence>
```

```
204         <xs:element name="KeybandId" minOccurs="0" type="xs:int"/>
205         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
206         <xs:element name="WeekInYearEnd" minOccurs="0" type="xs:int"/>
207         <xs:element name="WeekInYearStart" minOccurs="0" type="xs:int"/>
208         <xs:element name="YearEnd" minOccurs="0" type="xs:int"/>
209         <xs:element name="YearStart" minOccurs="0" type="xs:int"/>
210       </xs:sequence>
211     </xs:complexType>
212   </xs:element>
213   <xs:element name="ListKeysFromKeyBandResponse">
214     <xs:complexType>
215       <xs:sequence>
216         <xs:element xmlns:q5="http://schemas.datacontract.org/2004/07/KeyManagement" name="
                SharingKeyInfoList" minOccurs="0" nillable="true" type="q5:ArrayOfSharingKeyInfo"
                />
217         <xs:element name="Status" minOccurs="0" type="xs:int"/>
218         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
219       </xs:sequence>
220     </xs:complexType>
221   </xs:element>
222   <xs:element name="AddKeyAccessRequest">
223     <xs:complexType>
224       <xs:sequence>
225         <xs:element xmlns:q6="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
                "KeyIdList" minOccurs="0" nillable="true" type="q6:ArrayOfint"/>
226         <xs:element name="UsernameKeyOwner" minOccurs="0" nillable="true" type="xs:string"/>
227         <xs:element name="UsernameKeyRequester" minOccurs="0" nillable="true" type="xs:string"
                />
228       </xs:sequence>
229     </xs:complexType>
230   </xs:element>
231   <xs:element name="AddKeyAccessResponse">
232     <xs:complexType>
233       <xs:sequence>
234         <xs:element name="Status" minOccurs="0" type="xs:int"/>
235         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
236       </xs:sequence>
237     </xs:complexType>
238   </xs:element>
239   <xs:element name="DeleteShareRequestRequest">
240     <xs:complexType>
241       <xs:sequence>
242         <xs:element name="UsernameKeyOwner" minOccurs="0" nillable="true" type="xs:string"/>
243         <xs:element name="UsernameKeyRequester" minOccurs="0" nillable="true" type="xs:string"
                />
244       </xs:sequence>
245     </xs:complexType>
246   </xs:element>
247   <xs:element name="DeleteShareRequestResponse">
248     <xs:complexType>
249       <xs:sequence>
250         <xs:element name="Status" minOccurs="0" type="xs:int"/>
251         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
252       </xs:sequence>
253     </xs:complexType>
254   </xs:element>
255   <xs:element name="ListKeybandSharesRequest">
256     <xs:complexType>
257       <xs:sequence>
258         <xs:element name="KeyBandID" minOccurs="0" type="xs:int"/>
259         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
260       </xs:sequence>
261     </xs:complexType>
262   </xs:element>
263   <xs:element name="ListKeybandSharesResponse">
264     <xs:complexType>
265       <xs:sequence>
266         <xs:element xmlns:q7="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
                "KeyIdList" minOccurs="0" nillable="true" type="q7:ArrayOfint"/>
267         <xs:element name="Status" minOccurs="0" type="xs:int"/>
268         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
269       </xs:sequence>
270     </xs:complexType>
271   </xs:element>
272 </xs:schema>
```

http://localhost:8010/?xsd=xsd1

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/" attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/">
  <xs:element name="anyType" nillable="true" type="xs:anyType"/>
  <xs:element name="anyURI" nillable="true" type="xs:anyURI"/>
  <xs:element name="base64Binary" nillable="true" type="xs:base64Binary"/>
  <xs:element name="boolean" nillable="true" type="xs:boolean"/>
  <xs:element name="byte" nillable="true" type="xs:byte"/>
  <xs:element name="dateTime" nillable="true" type="xs:dateTime"/>
  <xs:element name="decimal" nillable="true" type="xs:decimal"/>
  <xs:element name="double" nillable="true" type="xs:double"/>
  <xs:element name="float" nillable="true" type="xs:float"/>
  <xs:element name="int" nillable="true" type="xs:int"/>
  <xs:element name="long" nillable="true" type="xs:long"/>
  <xs:element name="QName" nillable="true" type="xs:QName"/>
  <xs:element name="short" nillable="true" type="xs:short"/>
  <xs:element name="string" nillable="true" type="xs:string"/>
  <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte"/>
  <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt"/>
  <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong"/>
  <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort"/>
  <xs:element name="char" nillable="true" type="tns:char"/>
  <xs:simpleType name="char">
    <xs:restriction base="xs:int"/>
  </xs:simpleType>
  <xs:element name="duration" nillable="true" type="tns:duration"/>
  <xs:simpleType name="duration">
    <xs:restriction base="xs:duration">
      <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
      <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
      <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="guid" nillable="true" type="tns:guid"/>
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{12}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:attribute name="FactoryType" type="xs:QName"/>
  <xs:attribute name="Id" type="xs:ID"/>
  <xs:attribute name="Ref" type="xs:IDREF"/>
</xs:schema>
```

http://localhost:8010/?xsd=xsd2

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.datacontract.org/2004/07/KeyManagement" elementFormDefault="qualified" targetNamespace="http://schemas.datacontract.org/2004/07/KeyManagement">
  <xs:complexType name="ArrayOfdataStorage">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="dataStorage" nillable="true" type="tns:dataStorage"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfdataStorage" nillable="true" type="tns:ArrayOfdataStorage"/>
  <xs:complexType name="dataStorage">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
      <xs:element name="LinkAddress" nillable="true" type="xs:string"/>
      <xs:element name="Name" nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="dataStorage" nillable="true" type="tns:dataStorage"/>
  <xs:complexType name="ArrayOfKeyInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="KeyInfo" nillable="true" type="tns:KeyInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfKeyInfo" nillable="true" type="tns:ArrayOfKeyInfo"/>
  <xs:complexType name="KeyInfo">
    <xs:sequence>
      <xs:element name="KeyID" type="xs:int"/>
      <xs:element name="AccessPrivateKey" type="xs:boolean"/>
```

```
27        <xs:element name="AccessPublicKey" type="xs:boolean"/>
28        <xs:element name="Algorithm" nillable="true" type="xs:string"/>
29        <xs:element name="KeybandID" type="xs:int"/>
30        <xs:element name="WeekInYear" type="xs:int"/>
31        <xs:element name="Year" type="xs:int"/>
32        <xs:element name="DataStorageLink" nillable="true" type="xs:string"/>
33        <xs:element name="Purpose" nillable="true" type="xs:string"/>
34      </xs:sequence>
35    </xs:complexType>
36    <xs:element name="KeyInfo" nillable="true" type="tns:KeyInfo"/>
37    <xs:complexType name="ArrayOfSharingKeyInfo">
38      <xs:sequence>
39        <xs:element minOccurs="0" maxOccurs="unbounded" name="SharingKeyInfo" nillable="true"
              type="tns:SharingKeyInfo"/>
40      </xs:sequence>
41    </xs:complexType>
42    <xs:element name="ArrayOfSharingKeyInfo" nillable="true" type="tns:ArrayOfSharingKeyInfo"/>
43    <xs:complexType name="SharingKeyInfo">
44      <xs:sequence>
45        <xs:element name="KeyID" type="xs:int"/>
46        <xs:element name="KeybandID" type="xs:int"/>
47        <xs:element name="WeekInYear" type="xs:int"/>
48        <xs:element name="Year" type="xs:int"/>
49        <xs:element name="Purpose" nillable="true" type="xs:string"/>
50      </xs:sequence>
51    </xs:complexType>
52    <xs:element name="SharingKeyInfo" nillable="true" type="tns:SharingKeyInfo"/>
53  </xs:schema>
```

http://localhost:8010/?xsd=xsd3

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
        /2003/10/Serialization/Arrays" elementFormDefault="qualified" targetNamespace="http://
        schemas.microsoft.com/2003/10/Serialization/Arrays">
3     <xs:complexType name="ArrayOfint">
4       <xs:sequence>
5         <xs:element minOccurs="0" maxOccurs="unbounded" name="int" type="xs:int"/>
6       </xs:sequence>
7     </xs:complexType>
8     <xs:element name="ArrayOfint" type="tns:ArrayOfint" nillable="true"/>
9   </xs:schema>
```

A.7.2 Interface zwischen automatischen Datenquellen und Schlüsselverwaltung

http://localhost:8013/?wsdl

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.
        xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
        wss-wssecurity-utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:tns="http://tempuri.org
        /" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas
        .xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/
        addressing/policy" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas
        .microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsaw="http://www.w3.org/2006/05/addressing
        /wsdl" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:wsa10="http://www.w3.
        org/2005/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" name="
        KeyManagementForAutomaticMeterService" targetNamespace="http://tempuri.org/">
3     <wsp:Policy wsu:Id="wsHttpBinding_KeyManagementForAutomaticMeter_policy">
4       <wsp:ExactlyOne>
5         <wsp:All>
6           <sp:SymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
7             <wsp:Policy>
8               <sp:ProtectionToken>
9                 <wsp:Policy>
10                  <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws
                        /2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
11                    <wsp:Policy>
12                      <sp:RequireDerivedKeys/>
13                      <sp:BootstrapPolicy>
14                        <wsp:Policy>
```

```
15                          <sp:SignedParts>
16                            <sp:Body/>
17                            <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
                                "/>
18                            <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
19                            <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
20                            <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
21                            <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
22                            <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
23                            <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
24                          </sp:SignedParts>
25                          <sp:EncryptedParts>
26                            <sp:Body/>
27                          </sp:EncryptedParts>
28                          <sp:SymmetricBinding>
29                            <wsp:Policy>
30                              <sp:ProtectionToken>
31                                <wsp:Policy>
32                                  <sp:SpnegoContextToken sp:IncludeToken="http://schemas.xmlsoap
                                        .org/ws/2005/07/securitypolicy/IncludeToken/
                                        AlwaysToRecipient">
33                                    <wsp:Policy>
34                                      <sp:RequireDerivedKeys/>
35                                    </wsp:Policy>
36                                  </sp:SpnegoContextToken>
37                                </wsp:Policy>
38                              </sp:ProtectionToken>
39                              <sp:AlgorithmSuite>
40                                <wsp:Policy>
41                                  <sp:Basic256/>
42                                </wsp:Policy>
43                              </sp:AlgorithmSuite>
44                              <sp:Layout>
45                                <wsp:Policy>
46                                  <sp:Strict/>
47                                </wsp:Policy>
48                              </sp:Layout>
49                              <sp:IncludeTimestamp/>
50                              <sp:EncryptSignature/>
51                              <sp:OnlySignEntireHeadersAndBody/>
52                            </wsp:Policy>
53                          </sp:SymmetricBinding>
54                          <sp:Wss11>
55                            <wsp:Policy>
56                              <sp:MustSupportRefKeyIdentifier/>
57                              <sp:MustSupportRefIssuerSerial/>
58                              <sp:MustSupportRefThumbprint/>
59                              <sp:MustSupportRefEncryptedKey/>
60                            </wsp:Policy>
61                          </sp:Wss11>
62                          <sp:Trust10>
63                            <wsp:Policy>
64                              <sp:MustSupportIssuedTokens/>
65                              <sp:RequireClientEntropy/>
66                              <sp:RequireServerEntropy/>
67                            </wsp:Policy>
68                          </sp:Trust10>
69                        </wsp:Policy>
70                      </sp:BootstrapPolicy>
71                    </wsp:Policy>
72                  </sp:SecureConversationToken>
73                </wsp:Policy>
74              </sp:ProtectionToken>
75              <sp:AlgorithmSuite>
76                <wsp:Policy>
77                  <sp:Basic256/>
78                </wsp:Policy>
79              </sp:AlgorithmSuite>
80              <sp:Layout>
81                <wsp:Policy>
82                  <sp:Strict/>
83                </wsp:Policy>
```

```
84              </sp:Layout>
85              <sp:IncludeTimestamp/>
86              <sp:EncryptSignature/>
87              <sp:OnlySignEntireHeadersAndBody/>
88            </wsp:Policy>
89          </sp:SymmetricBinding>
90          <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
91            <wsp:Policy>
92              <sp:MustSupportRefKeyIdentifier/>
93              <sp:MustSupportRefIssuerSerial/>
94              <sp:MustSupportRefThumbprint/>
95              <sp:MustSupportRefEncryptedKey/>
96            </wsp:Policy>
97          </sp:Wss11>
98          <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
99            <wsp:Policy>
100             <sp:MustSupportIssuedTokens/>
101             <sp:RequireClientEntropy/>
102             <sp:RequireServerEntropy/>
103           </wsp:Policy>
104         </sp:Trust10>
105         <wsaw:UsingAddressing/>
106       </wsp:All>
107     </wsp:ExactlyOne>
108   </wsp:Policy>
109   <wsp:Policy wsu:Id="
          wsHttpBinding_KeyManagementForAutomaticMeter_GetPublicKeyFromKeybandForAutomaticMeter_Input_policy
          ">
110     <wsp:ExactlyOne>
111       <wsp:All>
112         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
113           <sp:Body/>
114           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
115           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
116           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
117           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
118           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
119           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
120           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
121         </sp:SignedParts>
122         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
123           <sp:Body/>
124         </sp:EncryptedParts>
125       </wsp:All>
126     </wsp:ExactlyOne>
127   </wsp:Policy>
128   <wsp:Policy wsu:Id="
          wsHttpBinding_KeyManagementForAutomaticMeter_GetPublicKeyFromKeybandForAutomaticMeter_output_policy
          ">
129     <wsp:ExactlyOne>
130       <wsp:All>
131         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
132           <sp:Body/>
133           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
134           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
135           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
136           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
137           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
138           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
139           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
140         </sp:SignedParts>
141         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
142           <sp:Body/>
143         </sp:EncryptedParts>
144       </wsp:All>
145     </wsp:ExactlyOne>
146   </wsp:Policy>
147   <wsdl:types>
148     <xsd:schema targetNamespace="http://tempuri.org/Imports">
149       <xsd:import schemaLocation="http://localhost:8013/?xsd=xsd0" namespace="http://tempuri.
            org/"/>
150       <xsd:import schemaLocation="http://localhost:8013/?xsd=xsd1" namespace="http://schemas.
            microsoft.com/2003/10/Serialization/"/>
151     </xsd:schema>
152   </wsdl:types>
153   <wsdl:message name="GetPublicKeyFromKeybandForAutomaticMeterRequest">
154     <wsdl:part name="parameters" element="tns:GetPublicKeyFromKeybandForAutomaticMeterRequest"
          />
```

```
155   </wsdl:message>
156   <wsdl:message name="GetPublicKeyFromKeybandForAutomaticMeterResponse">
157     <wsdl:part name="parameters" element="tns:GetPublicKeyFromKeybandForAutomaticMeterResponse
            "/>
158   </wsdl:message>
159   <wsdl:portType name="IKeyManagementForAutomaticMeterService">
160     <wsdl:operation name="GetPublicKeyFromKeybandForAutomaticMeter">
161       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementForAutomaticMeterService/
              GetPublicKeyFromKeybandForAutomaticMeter" name="
              GetPublicKeyFromKeybandForAutomaticMeterRequest" message="
              tns:GetPublicKeyFromKeybandForAutomaticMeterRequest"/>
162       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementForAutomaticMeterService/
              GetPublicKeyFromKeybandForAutomaticMeterResponse" name="
              GetPublicKeyFromKeybandForAutomaticMeterResponse" message="
              tns:GetPublicKeyFromKeybandForAutomaticMeterResponse"/>
163     </wsdl:operation>
164   </wsdl:portType>
165   <wsdl:binding name="wsHttpBinding_KeyManagementForAutomaticMeter" type="
          tns:IKeyManagementForAutomaticMeterService">
166     <wsp:PolicyReference URI="#wsHttpBinding_KeyManagementForAutomaticMeter_policy"/>
167     <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
168     <wsdl:operation name="GetPublicKeyFromKeybandForAutomaticMeter">
169       <soap12:operation soapAction="http://tempuri.org/IKeyManagementForAutomaticMeterService/
              GetPublicKeyFromKeybandForAutomaticMeter" style="document"/>
170       <wsdl:input name="GetPublicKeyFromKeybandForAutomaticMeterRequest">
171         <wsp:PolicyReference URI="#
                wsHttpBinding_KeyManagementForAutomaticMeter_GetPublicKeyFromKeybandForAutomaticMeter_Input_policy
                "/>
172         <soap12:body use="literal"/>
173       </wsdl:input>
174       <wsdl:output name="GetPublicKeyFromKeybandForAutomaticMeterResponse">
175         <wsp:PolicyReference URI="#
                wsHttpBinding_KeyManagementForAutomaticMeter_GetPublicKeyFromKeybandForAutomaticMeter_output_policy
                "/>
176         <soap12:body use="literal"/>
177       </wsdl:output>
178     </wsdl:operation>
179   </wsdl:binding>
180   <wsdl:service name="KeyManagementForAutomaticMeterService">
181     <wsdl:port name="wsHttpBinding_KeyManagementForAutomaticMeter" binding="
            tns:wsHttpBinding_KeyManagementForAutomaticMeter">
182       <soap12:address location="http://localhost:8013/KeyManagementForAutomaticMeter"/>
183       <wsa10:EndpointReference>
184         <wsa10:Address>http://localhost:8013/KeyManagementForAutomaticMeter</wsa10:Address>
185         <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
186           <Upn>lucknerw@pdts.at</Upn>
187         </Identity>
188       </wsa10:EndpointReference>
189     </wsdl:port>
190   </wsdl:service>
191 </wsdl:definitions>
```

http://localhost:8013/?xsd=xsd0

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
        elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3     <xs:element name="GetPublicKeyFromKeybandForAutomaticMeterRequest">
4       <xs:complexType>
5         <xs:sequence>
6           <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
7           <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
8           <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
9           <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
10        </xs:sequence>
11      </xs:complexType>
12    </xs:element>
13    <xs:element name="GetPublicKeyFromKeybandForAutomaticMeterResponse">
14      <xs:complexType>
15        <xs:sequence>
16          <xs:element name="Algorithm" minOccurs="0" nillable="true" type="xs:string"/>
17          <xs:element name="KeyId" minOccurs="0" type="xs:int"/>
18          <xs:element name="PublicKey" minOccurs="0" nillable="true" type="xs:string"/>
19          <xs:element name="Status" minOccurs="0" type="xs:int"/>
20          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
21        </xs:sequence>
22      </xs:complexType>
23    </xs:element>
```

```
24 </xs:schema>
```

http://localhost:8013/?xsd=xsd1

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
       /2003/10/Serialization/" attributeFormDefault="qualified" elementFormDefault="qualified"
       targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/">
3    <xs:element name="anyType" nillable="true" type="xs:anyType"/>
4    <xs:element name="anyURI" nillable="true" type="xs:anyURI"/>
5    <xs:element name="base64Binary" nillable="true" type="xs:base64Binary"/>
6    <xs:element name="boolean" nillable="true" type="xs:boolean"/>
7    <xs:element name="byte" nillable="true" type="xs:byte"/>
8    <xs:element name="dateTime" nillable="true" type="xs:dateTime"/>
9    <xs:element name="decimal" nillable="true" type="xs:decimal"/>
10   <xs:element name="double" nillable="true" type="xs:double"/>
11   <xs:element name="float" nillable="true" type="xs:float"/>
12   <xs:element name="int" nillable="true" type="xs:int"/>
13   <xs:element name="long" nillable="true" type="xs:long"/>
14   <xs:element name="QName" nillable="true" type="xs:QName"/>
15   <xs:element name="short" nillable="true" type="xs:short"/>
16   <xs:element name="string" nillable="true" type="xs:string"/>
17   <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte"/>
18   <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt"/>
19   <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong"/>
20   <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort"/>
21   <xs:element name="char" nillable="true" type="tns:char"/>
22   <xs:simpleType name="char">
23     <xs:restriction base="xs:int"/>
24   </xs:simpleType>
25   <xs:element name="duration" nillable="true" type="tns:duration"/>
26   <xs:simpleType name="duration">
27     <xs:restriction base="xs:duration">
28       <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
29       <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
30       <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
31     </xs:restriction>
32   </xs:simpleType>
33   <xs:element name="guid" nillable="true" type="tns:guid"/>
34   <xs:simpleType name="guid">
35     <xs:restriction base="xs:string">
36       <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F
             ]{12}"/>
37     </xs:restriction>
38   </xs:simpleType>
39   <xs:attribute name="FactoryType" type="xs:QName"/>
40   <xs:attribute name="Id" type="xs:ID"/>
41   <xs:attribute name="Ref" type="xs:IDREF"/>
42 </xs:schema>
```

GetPublicKeyFromKeybandXmlRequest.xsd

```
1  ï»¿<?xml version="1.0" encoding="utf-8"?>
2  <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="GetPublicKeyFromKeybandXmlRequest" nillable="true" type="
         GetPublicKeyFromKeybandXmlRequest" />
4    <xs:complexType name="GetPublicKeyFromKeybandXmlRequest">
5      <xs:complexContent mixed="false">
6        <xs:extension base="GetPublicKeyFromKeybandRequest" />
7      </xs:complexContent>
8    </xs:complexType>
9    <xs:complexType name="GetPublicKeyFromKeybandRequest">
10     <xs:sequence>
11       <xs:element minOccurs="0" maxOccurs="1" name="DataStorageLink" type="xs:string" />
12       <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
13       <xs:element minOccurs="1" maxOccurs="1" name="KeybandId" type="xs:int" />
14       <xs:element minOccurs="1" maxOccurs="1" name="WeekInYear" type="xs:int" />
15       <xs:element minOccurs="1" maxOccurs="1" name="Year" type="xs:int" />
16     </xs:sequence>
17   </xs:complexType>
18 </xs:schema>
```

A.7.3 Interface zwischen Datenspeicher und Schlüsselverwaltung

http://localhost:8016/?wsdl

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.
       xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
       wss-wssecurity-utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:tns="http://tempuri.org
       /" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas
       .xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/
       addressing/policy" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas
       .microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsaw="http://www.w3.org/2006/05/addressing
       /wsdl" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:wsa10="http://www.w3.
       org/2005/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" name="
       KeyManagementForDataStorageService" targetNamespace="http://tempuri.org/">
3    <wsp:Policy wsu:Id="wsHttpBinding_KeyManagementForDataStorage_policy">
4      <wsp:ExactlyOne>
5        <wsp:All>
6          <sp:SymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
7            <wsp:Policy>
8              <sp:ProtectionToken>
9                <wsp:Policy>
10                 <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws
                       /2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
11                   <wsp:Policy>
12                     <sp:RequireDerivedKeys/>
13                     <sp:BootstrapPolicy>
14                       <wsp:Policy>
15                         <sp:SignedParts>
16                           <sp:Body/>
17                           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
                                 "/>
18                           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
19                           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
20                           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
21                           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
22                           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
23                           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/
                                 addressing"/>
24                         </sp:SignedParts>
25                         <sp:EncryptedParts>
26                           <sp:Body/>
27                         </sp:EncryptedParts>
28                         <sp:SymmetricBinding>
29                           <wsp:Policy>
30                             <sp:ProtectionToken>
31                               <wsp:Policy>
32                                 <sp:SpnegoContextToken sp:IncludeToken="http://schemas.xmlsoap
                                       .org/ws/2005/07/securitypolicy/IncludeToken/
                                       AlwaysToRecipient">
33                                   <wsp:Policy>
34                                     <sp:RequireDerivedKeys/>
35                                   </wsp:Policy>
36                                 </sp:SpnegoContextToken>
37                               </wsp:Policy>
38                             </sp:ProtectionToken>
39                             <sp:AlgorithmSuite>
40                               <wsp:Policy>
41                                 <sp:Basic256/>
42                               </wsp:Policy>
43                             </sp:AlgorithmSuite>
44                             <sp:Layout>
45                               <wsp:Policy>
46                                 <sp:Strict/>
47                               </wsp:Policy>
48                             </sp:Layout>
49                             <sp:IncludeTimestamp/>
50                             <sp:EncryptSignature/>
51                             <sp:OnlySignEntireHeadersAndBody/>
52                           </wsp:Policy>
53                         </sp:SymmetricBinding>
54                         <sp:Wss11>
55                           <wsp:Policy>
56                             <sp:MustSupportRefKeyIdentifier/>
57                             <sp:MustSupportRefIssuerSerial/>
58                             <sp:MustSupportRefThumbprint/>
```

```
59                              <sp:MustSupportRefEncryptedKey/>
60                            </wsp:Policy>
61                          </sp:Wss11>
62                          <sp:Trust10>
63                            <wsp:Policy>
64                              <sp:MustSupportIssuedTokens/>
65                              <sp:RequireClientEntropy/>
66                              <sp:RequireServerEntropy/>
67                            </wsp:Policy>
68                          </sp:Trust10>
69                        </wsp:Policy>
70                      </sp:BootstrapPolicy>
71                    </wsp:Policy>
72                  </sp:SecureConversationToken>
73                </wsp:Policy>
74              </sp:ProtectionToken>
75              <sp:AlgorithmSuite>
76                <wsp:Policy>
77                  <sp:Basic256/>
78                </wsp:Policy>
79              </sp:AlgorithmSuite>
80              <sp:Layout>
81                <wsp:Policy>
82                  <sp:Strict/>
83                </wsp:Policy>
84              </sp:Layout>
85              <sp:IncludeTimestamp/>
86              <sp:EncryptSignature/>
87              <sp:OnlySignEntireHeadersAndBody/>
88            </wsp:Policy>
89          </sp:SymmetricBinding>
90          <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
91            <wsp:Policy>
92              <sp:MustSupportRefKeyIdentifier/>
93              <sp:MustSupportRefIssuerSerial/>
94              <sp:MustSupportRefThumbprint/>
95              <sp:MustSupportRefEncryptedKey/>
96            </wsp:Policy>
97          </sp:Wss11>
98          <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
99            <wsp:Policy>
100             <sp:MustSupportIssuedTokens/>
101             <sp:RequireClientEntropy/>
102             <sp:RequireServerEntropy/>
103           </wsp:Policy>
104         </sp:Trust10>
105         <wsaw:UsingAddressing/>
106       </wsp:All>
107     </wsp:ExactlyOne>
108   </wsp:Policy>
109   <wsp:Policy wsu:Id="
          wsHttpBinding_KeyManagementForDataStorage_GetPublicKeyForSignature_Input_policy">
110     <wsp:ExactlyOne>
111       <wsp:All>
112         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
113           <sp:Body/>
114           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
115           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
116           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
117           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
118           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
119           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
120           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
121         </sp:SignedParts>
122         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
123           <sp:Body/>
124         </sp:EncryptedParts>
125       </wsp:All>
126     </wsp:ExactlyOne>
127   </wsp:Policy>
128   <wsp:Policy wsu:Id="
          wsHttpBinding_KeyManagementForDataStorage_GetPublicKeyForSignature_output_policy">
129     <wsp:ExactlyOne>
130       <wsp:All>
131         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
132           <sp:Body/>
133           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
134           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
135           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
136           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
137           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
138           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
139           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
140         </sp:SignedParts>
141         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
142           <sp:Body/>
143         </sp:EncryptedParts>
144       </wsp:All>
145     </wsp:ExactlyOne>
146   </wsp:Policy>
147   <wsdl:types>
148     <xsd:schema targetNamespace="http://tempuri.org/Imports">
149       <xsd:import schemaLocation="http://localhost:8016/?xsd=xsd0" namespace="http://tempuri.
              org/"/>
150       <xsd:import schemaLocation="http://localhost:8016/?xsd=xsd1" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/"/>
151     </xsd:schema>
152   </wsdl:types>
153   <wsdl:message name="GetPublicKeyForSignatureRequest">
154     <wsdl:part name="parameters" element="tns:GetPublicKeyForSignatureRequest"/>
155   </wsdl:message>
156   <wsdl:message name="GetPublicKeyForSignatureResponse">
157     <wsdl:part name="parameters" element="tns:GetPublicKeyForSignatureResponse"/>
158   </wsdl:message>
159   <wsdl:portType name="IKeyManagementForDataStorageService">
160     <wsdl:operation name="GetPublicKeyForSignature">
161       <wsdl:input wsaw:Action="http://tempuri.org/IKeyManagementForDataStorageService/
              GetPublicKeyForSignature" name="GetPublicKeyForSignatureRequest" message="
              tns:GetPublicKeyForSignatureRequest"/>
162       <wsdl:output wsaw:Action="http://tempuri.org/IKeyManagementForDataStorageService/
              GetPublicKeyForSignatureResponse" name="GetPublicKeyForSignatureResponse" message="
              tns:GetPublicKeyForSignatureResponse"/>
163     </wsdl:operation>
164   </wsdl:portType>
165   <wsdl:binding name="wsHttpBinding_KeyManagementForDataStorage" type="
        tns:IKeyManagementForDataStorageService">
166     <wsp:PolicyReference URI="#wsHttpBinding_KeyManagementForDataStorage_policy"/>
167     <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
168     <wsdl:operation name="GetPublicKeyForSignature">
169       <soap12:operation soapAction="http://tempuri.org/IKeyManagementForDataStorageService/
              GetPublicKeyForSignature" style="document"/>
170       <wsdl:input name="GetPublicKeyForSignatureRequest">
171         <wsp:PolicyReference URI="#
                wsHttpBinding_KeyManagementForDataStorage_GetPublicKeyForSignature_Input_policy"/
                >
172         <soap12:body use="literal"/>
173       </wsdl:input>
174       <wsdl:output name="GetPublicKeyForSignatureResponse">
175         <wsp:PolicyReference URI="#
                wsHttpBinding_KeyManagementForDataStorage_GetPublicKeyForSignature_output_policy"
                />
176         <soap12:body use="literal"/>
177       </wsdl:output>
178     </wsdl:operation>
179   </wsdl:binding>
180   <wsdl:service name="KeyManagementForDataStorageService">
181     <wsdl:port name="wsHttpBinding_KeyManagementForDataStorage" binding="
          tns:wsHttpBinding_KeyManagementForDataStorage">
182       <soap12:address location="http://localhost:8016/KeyManagementForDataStorage"/>
183       <wsa10:EndpointReference>
184         <wsa10:Address>http://localhost:8016/KeyManagementForDataStorage</wsa10:Address>
185         <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
186           <Upn>lucknerw@pdts.at</Upn>
187         </Identity>
188       </wsa10:EndpointReference>
189     </wsdl:port>
190   </wsdl:service>
191 </wsdl:definitions>
```

http://localhost:8016/?xsd=xsd0

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
       elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3    <xs:element name="GetPublicKeyForSignatureRequest">
4      <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="SignauterKeyId" minOccurs="0" type="xs:int"/>
        <xs:element name="Username" minOccurs="0" type="xs:string" nillable="true"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetPublicKeyForSignatureResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Algorithm" minOccurs="0" type="xs:string" nillable="true"/>
        <xs:element name="PublicKey" minOccurs="0" type="xs:string" nillable="true"/>
        <xs:element name="Status" minOccurs="0" type="xs:int"/>
        <xs:element name="StatusMessage" minOccurs="0" type="xs:string" nillable="true"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

http://localhost:8016/?xsd=xsd1

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
    /2003/10/Serialization/" attributeFormDefault="qualified" elementFormDefault="qualified"
    targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/">
  <xs:element name="anyType" nillable="true" type="xs:anyType"/>
  <xs:element name="anyURI" nillable="true" type="xs:anyURI"/>
  <xs:element name="base64Binary" nillable="true" type="xs:base64Binary"/>
  <xs:element name="boolean" nillable="true" type="xs:boolean"/>
  <xs:element name="byte" nillable="true" type="xs:byte"/>
  <xs:element name="dateTime" nillable="true" type="xs:dateTime"/>
  <xs:element name="decimal" nillable="true" type="xs:decimal"/>
  <xs:element name="double" nillable="true" type="xs:double"/>
  <xs:element name="float" nillable="true" type="xs:float"/>
  <xs:element name="int" nillable="true" type="xs:int"/>
  <xs:element name="long" nillable="true" type="xs:long"/>
  <xs:element name="QName" nillable="true" type="xs:QName"/>
  <xs:element name="short" nillable="true" type="xs:short"/>
  <xs:element name="string" nillable="true" type="xs:string"/>
  <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte"/>
  <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt"/>
  <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong"/>
  <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort"/>
  <xs:element name="char" nillable="true" type="tns:char"/>
  <xs:simpleType name="char">
    <xs:restriction base="xs:int"/>
  </xs:simpleType>
  <xs:element name="duration" nillable="true" type="tns:duration"/>
  <xs:simpleType name="duration">
    <xs:restriction base="xs:duration">
      <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
      <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
      <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="guid" nillable="true" type="tns:guid"/>
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F
          ]{12}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:attribute name="FactoryType" type="xs:QName"/>
  <xs:attribute name="Id" type="xs:ID"/>
  <xs:attribute name="Ref" type="xs:IDREF"/>
</xs:schema>
```

A.7.4 Interface zwischen Client und Datenspeicher

http://localhost:8011/?wsdl

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.
    xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
    wss-wssecurity-utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
      xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:tns="http://tempuri.org
      /" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas
      .xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/
      addressing/policy" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas
      .microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsaw="http://www.w3.org/2006/05/addressing
      /wsdl" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:wsa10="http://www.w3.
      org/2005/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" name="
      DataStorageService" targetNamespace="http://tempuri.org/">
3   <wsp:Policy wsu:Id="wsDualHttpBinding_policy">
4     <wsp:ExactlyOne>
5       <wsp:All>
6         <wsrm:RMAssertion xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy">
7           <wsrm:InactivityTimeout Milliseconds="600000"/>
8           <wsrm:AcknowledgementInterval Milliseconds="200"/>
9         </wsrm:RMAssertion>
10        <sp:SymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
11          <wsp:Policy>
12            <sp:ProtectionToken>
13              <wsp:Policy>
14                <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws
                      /2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
15                  <wsp:Policy>
16                    <sp:RequireDerivedKeys/>
17                    <sp:BootstrapPolicy>
18                      <wsp:Policy>
19                        <sp:SignedParts>
20                          <sp:Body/>
21                          <sp:Header Name="ChannelInstance" Namespace="http://schemas.
                                microsoft.com/ws/2005/02/duplex"/>
22                          <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
                                "/>
23                          <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
24                          <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
25                          <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
26                          <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
27                          <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
28                          <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/
                                addressing"/>
29                        </sp:SignedParts>
30                        <sp:EncryptedParts>
31                          <sp:Body/>
32                        </sp:EncryptedParts>
33                        <sp:SymmetricBinding>
34                          <wsp:Policy>
35                            <sp:ProtectionToken>
36                              <wsp:Policy>
37                                <sp:SpnegoContextToken sp:IncludeToken="http://schemas.xmlsoap
                                      .org/ws/2005/07/securitypolicy/IncludeToken/
                                      AlwaysToRecipient">
38                                  <wsp:Policy>
39                                    <sp:RequireDerivedKeys/>
40                                  </wsp:Policy>
41                                </sp:SpnegoContextToken>
42                              </wsp:Policy>
43                            </sp:ProtectionToken>
44                            <sp:AlgorithmSuite>
45                              <wsp:Policy>
46                                <sp:Basic256/>
47                              </wsp:Policy>
48                            </sp:AlgorithmSuite>
49                            <sp:Layout>
50                              <wsp:Policy>
51                                <sp:Strict/>
52                              </wsp:Policy>
53                            </sp:Layout>
54                            <sp:IncludeTimestamp/>
55                            <sp:EncryptSignature/>
56                            <sp:OnlySignEntireHeadersAndBody/>
57                          </wsp:Policy>
58                        </sp:SymmetricBinding>
59                        <sp:Wss11>
60                          <wsp:Policy>
61                            <sp:MustSupportRefKeyIdentifier/>
```

```
62                              <sp:MustSupportRefIssuerSerial/>
63                              <sp:MustSupportRefThumbprint/>
64                              <sp:MustSupportRefEncryptedKey/>
65                            </wsp:Policy>
66                          </sp:Wss11>
67                          <sp:Trust10>
68                            <wsp:Policy>
69                              <sp:MustSupportIssuedTokens/>
70                              <sp:RequireClientEntropy/>
71                              <sp:RequireServerEntropy/>
72                            </wsp:Policy>
73                          </sp:Trust10>
74                        </wsp:Policy>
75                      </sp:BootstrapPolicy>
76                    </wsp:Policy>
77                  </sp:SecureConversationToken>
78                </wsp:Policy>
79              </sp:ProtectionToken>
80              <sp:AlgorithmSuite>
81                <wsp:Policy>
82                  <sp:Basic256/>
83                </wsp:Policy>
84              </sp:AlgorithmSuite>
85              <sp:Layout>
86                <wsp:Policy>
87                  <sp:Strict/>
88                </wsp:Policy>
89              </sp:Layout>
90              <sp:IncludeTimestamp/>
91              <sp:EncryptSignature/>
92              <sp:OnlySignEntireHeadersAndBody/>
93            </wsp:Policy>
94          </sp:SymmetricBinding>
95          <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
96            <wsp:Policy>
97              <sp:MustSupportRefKeyIdentifier/>
98              <sp:MustSupportRefIssuerSerial/>
99              <sp:MustSupportRefThumbprint/>
100             <sp:MustSupportRefEncryptedKey/>
101           </wsp:Policy>
102         </sp:Wss11>
103         <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
104           <wsp:Policy>
105             <sp:MustSupportIssuedTokens/>
106             <sp:RequireClientEntropy/>
107             <sp:RequireServerEntropy/>
108           </wsp:Policy>
109         </sp:Trust10>
110         <cdp:CompositeDuplex xmlns:cdp="http://schemas.microsoft.com/net/2006/06/duplex"/>
111         <ow:OneWay xmlns:ow="http://schemas.microsoft.com/ws/2005/05/routing/policy"/>
112         <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/
                policy/optimizedmimeserialization"/>
113         <wsaw:UsingAddressing/>
114       </wsp:All>
115     </wsp:ExactlyOne>
116   </wsp:Policy>
117   <wsp:Policy wsu:Id="wsDualHttpBinding_AddNewUser_Input_policy">
118     <wsp:ExactlyOne>
119       <wsp:All>
120         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
121           <sp:Body/>
122           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
123           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
124           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
125           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
126           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
127           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
128           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
129           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
130           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
131           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
132           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
133         </sp:SignedParts>
134         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
135           <sp:Body/>
```

```
136         </sp:EncryptedParts>
137       </wsp:All>
138     </wsp:ExactlyOne>
139   </wsp:Policy>
140   <wsp:Policy wsu:Id="wsDualHttpBinding_AddNewUser_output_policy">
141     <wsp:ExactlyOne>
142       <wsp:All>
143         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
144           <sp:Body/>
145           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
146           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
147           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
148           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
149           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
150           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
151           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
152           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
153           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
154           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
155           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
156         </sp:SignedParts>
157         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
158           <sp:Body/>
159         </sp:EncryptedParts>
160       </wsp:All>
161     </wsp:ExactlyOne>
162   </wsp:Policy>
163   <wsp:Policy wsu:Id="wsDualHttpBinding_ListData_Input_policy">
164     <wsp:ExactlyOne>
165       <wsp:All>
166         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
167           <sp:Body/>
168           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
169           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
170           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
171           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
172           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
173           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
174           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
175           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
176           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
177           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
178           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
179         </sp:SignedParts>
180         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
181           <sp:Body/>
182         </sp:EncryptedParts>
183       </wsp:All>
184     </wsp:ExactlyOne>
185   </wsp:Policy>
186   <wsp:Policy wsu:Id="wsDualHttpBinding_ListData_output_policy">
187     <wsp:ExactlyOne>
188       <wsp:All>
189         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
190           <sp:Body/>
191           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
192           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
193           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
194           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
195           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
196           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
197           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
198           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
199           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
200           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
201           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
202         </sp:SignedParts>
203         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
204           <sp:Body/>
```

```
205         </sp:EncryptedParts>
206       </wsp:All>
207     </wsp:ExactlyOne>
208   </wsp:Policy>
209   <wsp:Policy wsu:Id="wsDualHttpBinding_GetData_Input_policy">
210     <wsp:ExactlyOne>
211       <wsp:All>
212         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
213           <sp:Body/>
214           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
215           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
216           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
217           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
218           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
219           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
220           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
221           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
222           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
223           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
224           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
225         </sp:SignedParts>
226         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
227           <sp:Body/>
228         </sp:EncryptedParts>
229       </wsp:All>
230     </wsp:ExactlyOne>
231   </wsp:Policy>
232   <wsp:Policy wsu:Id="wsDualHttpBinding_GetData_output_policy">
233     <wsp:ExactlyOne>
234       <wsp:All>
235         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
236           <sp:Body/>
237           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
238           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
239           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
240           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
241           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
242           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
243           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
244           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
245           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
246           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
247           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
248         </sp:SignedParts>
249         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
250           <sp:Body/>
251         </sp:EncryptedParts>
252       </wsp:All>
253     </wsp:ExactlyOne>
254   </wsp:Policy>
255   <wsp:Policy wsu:Id="wsDualHttpBinding_GetNewIds_Input_policy">
256     <wsp:ExactlyOne>
257       <wsp:All>
258         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
259           <sp:Body/>
260           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
261           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
262           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
263           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
264           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
265           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
266           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
267           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
268           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
269           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
270           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
271         </sp:SignedParts>
272         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
273           <sp:Body/>
```

```
274         </sp:EncryptedParts>
275       </wsp:All>
276     </wsp:ExactlyOne>
277   </wsp:Policy>
278   <wsp:Policy wsu:Id="wsDualHttpBinding_GetNewIds_output_policy">
279     <wsp:ExactlyOne>
280       <wsp:All>
281         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
282           <sp:Body/>
283           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
284           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
285           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
286           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
287           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
288           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
289           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
290           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
291           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
292           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
293           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
294         </sp:SignedParts>
295         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
296           <sp:Body/>
297         </sp:EncryptedParts>
298       </wsp:All>
299     </wsp:ExactlyOne>
300   </wsp:Policy>
301   <wsp:Policy wsu:Id="wsDualHttpBinding_AddUpdateData_Input_policy">
302     <wsp:ExactlyOne>
303       <wsp:All>
304         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
305           <sp:Body/>
306           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
307           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
308           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
309           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
310           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
311           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
312           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
313           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
314           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
315           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
316           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
317         </sp:SignedParts>
318         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
319           <sp:Body/>
320         </sp:EncryptedParts>
321       </wsp:All>
322     </wsp:ExactlyOne>
323   </wsp:Policy>
324   <wsp:Policy wsu:Id="wsDualHttpBinding_AddUpdateData_output_policy">
325     <wsp:ExactlyOne>
326       <wsp:All>
327         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
328           <sp:Body/>
329           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
330           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
331           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
332           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
333           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
334           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
335           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
336           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
337           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
338           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
339           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
340         </sp:SignedParts>
341         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
342           <sp:Body/>
```

```
343         </sp:EncryptedParts>
344       </wsp:All>
345     </wsp:ExactlyOne>
346   </wsp:Policy>
347   <wsp:Policy wsu:Id="wsDualHttpBinding_SubscribeLock_Input_policy">
348     <wsp:ExactlyOne>
349       <wsp:All>
350         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
351           <sp:Body/>
352           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
353           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
354           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
355           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
356           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
357           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
358           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
359           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
360           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
361           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
362           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
363         </sp:SignedParts>
364         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
365           <sp:Body/>
366         </sp:EncryptedParts>
367       </wsp:All>
368     </wsp:ExactlyOne>
369   </wsp:Policy>
370   <wsp:Policy wsu:Id="wsDualHttpBinding_SubscribeLock_output_policy">
371     <wsp:ExactlyOne>
372       <wsp:All>
373         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
374           <sp:Body/>
375           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
376           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
377           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
378           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
379           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
380           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
381           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
382           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
383           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
384           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
385           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
386         </sp:SignedParts>
387         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
388           <sp:Body/>
389         </sp:EncryptedParts>
390       </wsp:All>
391     </wsp:ExactlyOne>
392   </wsp:Policy>
393   <wsp:Policy wsu:Id="wsDualHttpBinding_ReleaseLock_Input_policy">
394     <wsp:ExactlyOne>
395       <wsp:All>
396         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
397           <sp:Body/>
398           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
399           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
400           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
401           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
402           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
403           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
404           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
405           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
406           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
407           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
408           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
409         </sp:SignedParts>
410         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
411           <sp:Body/>
```

```
412         </sp:EncryptedParts>
413       </wsp:All>
414     </wsp:ExactlyOne>
415   </wsp:Policy>
416   <wsp:Policy wsu:Id="wsDualHttpBinding_ReleaseLock_output_policy">
417     <wsp:ExactlyOne>
418       <wsp:All>
419         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
420           <sp:Body/>
421           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
422           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
423           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
424           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
425           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
426           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
427           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
428           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
429           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
430           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
431           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
432         </sp:SignedParts>
433         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
434           <sp:Body/>
435         </sp:EncryptedParts>
436       </wsp:All>
437     </wsp:ExactlyOne>
438   </wsp:Policy>
439   <wsp:Policy wsu:Id="wsDualHttpBinding_NotifyLock_output_policy">
440     <wsp:ExactlyOne>
441       <wsp:All>
442         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
443           <sp:Body/>
444           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
445           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
446           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
447           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
448           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
449           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
450           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
451           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
452           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
453           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
454           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
455         </sp:SignedParts>
456         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
457           <sp:Body/>
458         </sp:EncryptedParts>
459       </wsp:All>
460     </wsp:ExactlyOne>
461   </wsp:Policy>
462   <wsdl:types>
463     <xsd:schema targetNamespace="http://tempuri.org/Imports">
464       <xsd:import schemaLocation="http://localhost:8011/?xsd=xsd0" namespace="http://tempuri.
              org/"/>
465       <xsd:import schemaLocation="http://localhost:8011/?xsd=xsd1" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/"/>
466       <xsd:import schemaLocation="http://localhost:8011/?xsd=xsd2" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/Arrays"/>
467       <xsd:import schemaLocation="http://localhost:8011/?xsd=xsd3" namespace="http://schemas.
              datacontract.org/2004/07/DataStorage"/>
468     </xsd:schema>
469   </wsdl:types>
470   <wsdl:message name="AddNewUserReqeust">
471     <wsdl:part name="parameters" element="tns:AddNewUserReqeust"/>
472   </wsdl:message>
473   <wsdl:message name="AddNewUserResponse">
474     <wsdl:part name="parameters" element="tns:AddNewUserResponse"/>
475   </wsdl:message>
476   <wsdl:message name="ListDataRequest">
477     <wsdl:part name="parameters" element="tns:ListDataRequest"/>
478   </wsdl:message>
479   <wsdl:message name="ListDataResponse">
```

```
480     <wsdl:part name="parameters" element="tns:ListDataResponse"/>
481   </wsdl:message>
482   <wsdl:message name="GetDataRequest">
483     <wsdl:part name="parameters" element="tns:GetDataRequest"/>
484   </wsdl:message>
485   <wsdl:message name="GetDataResponse">
486     <wsdl:part name="parameters" element="tns:GetDataResponse"/>
487   </wsdl:message>
488   <wsdl:message name="GetNewIdsRequest">
489     <wsdl:part name="parameters" element="tns:GetNewIdsRequest"/>
490   </wsdl:message>
491   <wsdl:message name="GetNewIdsResponse">
492     <wsdl:part name="parameters" element="tns:GetNewIdsResponse"/>
493   </wsdl:message>
494   <wsdl:message name="AddUpdateDataRequest">
495     <wsdl:part name="parameters" element="tns:AddUpdateDataRequest"/>
496   </wsdl:message>
497   <wsdl:message name="AddUpdateDataResponse">
498     <wsdl:part name="parameters" element="tns:AddUpdateDataResponse"/>
499   </wsdl:message>
500   <wsdl:message name="SubscribeLockRequest">
501     <wsdl:part name="parameters" element="tns:SubscribeLockRequest"/>
502   </wsdl:message>
503   <wsdl:message name="SubscribeLockResponse">
504     <wsdl:part name="parameters" element="tns:SubscribeLockResponse"/>
505   </wsdl:message>
506   <wsdl:message name="ReleaseLockRequest">
507     <wsdl:part name="parameters" element="tns:ReleaseLockRequest"/>
508   </wsdl:message>
509   <wsdl:message name="ReleaseLockResponse">
510     <wsdl:part name="parameters" element="tns:ReleaseLockResponse"/>
511   </wsdl:message>
512   <wsdl:message name="NotifyLockNotification">
513     <wsdl:part name="parameters" element="tns:NotifyLockNotification"/>
514   </wsdl:message>
515   <wsdl:portType name="IDataStorageService">
516     <wsdl:operation name="AddNewUser">
517       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/AddNewUser" name="
              AddNewUserReqeust" message="tns:AddNewUserReqeust"/>
518       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/AddNewUserResponse"
              name="AddNewUserResponse" message="tns:AddNewUserResponse"/>
519     </wsdl:operation>
520     <wsdl:operation name="ListData">
521       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/ListData" name="
              ListDataRequest" message="tns:ListDataRequest"/>
522       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/ListDataResponse" name=
              "ListDataResponse" message="tns:ListDataResponse"/>
523     </wsdl:operation>
524     <wsdl:operation name="GetData">
525       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/GetData" name="
              GetDataRequest" message="tns:GetDataRequest"/>
526       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/GetDataResponse" name="
              GetDataResponse" message="tns:GetDataResponse"/>
527     </wsdl:operation>
528     <wsdl:operation name="GetNewIds">
529       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/GetNewIds" name="
              GetNewIdsRequest" message="tns:GetNewIdsRequest"/>
530       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/GetNewIdsResponse" name
              ="GetNewIdsResponse" message="tns:GetNewIdsResponse"/>
531     </wsdl:operation>
532     <wsdl:operation name="AddUpdateData">
533       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/AddUpdateData" name="
              AddUpdateDataRequest" message="tns:AddUpdateDataRequest"/>
534       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/AddUpdateDataResponse"
              name="AddUpdateDataResponse" message="tns:AddUpdateDataResponse"/>
535     </wsdl:operation>
536     <wsdl:operation name="SubscribeLock">
537       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/SubscribeLock" name="
              SubscribeLockRequest" message="tns:SubscribeLockRequest"/>
538       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/SubscribeLockResponse"
              name="SubscribeLockResponse" message="tns:SubscribeLockResponse"/>
539     </wsdl:operation>
540     <wsdl:operation name="ReleaseLock">
541       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageService/ReleaseLock" name="
              ReleaseLockRequest" message="tns:ReleaseLockRequest"/>
542       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/ReleaseLockResponse"
              name="ReleaseLockResponse" message="tns:ReleaseLockResponse"/>
543     </wsdl:operation>
```

```
544       <wsdl:operation name="NotifyLock">
545         <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageService/NotifyLock" name="
                NotifyLockNotification" message="tns:NotifyLockNotification"/>
546       </wsdl:operation>
547     </wsdl:portType>
548     <wsdl:binding name="wsDualHttpBinding" type="tns:IDataStorageService">
549       <wsp:PolicyReference URI="#wsDualHttpBinding_policy"/>
550       <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
551       <wsdl:operation name="AddNewUser">
552         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/AddNewUser" style="
                document"/>
553         <wsdl:input name="AddNewUserReqeust">
554           <wsp:PolicyReference URI="#wsDualHttpBinding_AddNewUser_Input_policy"/>
555           <soap12:body use="literal"/>
556         </wsdl:input>
557         <wsdl:output name="AddNewUserResponse">
558           <wsp:PolicyReference URI="#wsDualHttpBinding_AddNewUser_output_policy"/>
559           <soap12:body use="literal"/>
560         </wsdl:output>
561       </wsdl:operation>
562       <wsdl:operation name="ListData">
563         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/ListData" style="
                document"/>
564         <wsdl:input name="ListDataRequest">
565           <wsp:PolicyReference URI="#wsDualHttpBinding_ListData_Input_policy"/>
566           <soap12:body use="literal"/>
567         </wsdl:input>
568         <wsdl:output name="ListDataResponse">
569           <wsp:PolicyReference URI="#wsDualHttpBinding_ListData_output_policy"/>
570           <soap12:body use="literal"/>
571         </wsdl:output>
572       </wsdl:operation>
573       <wsdl:operation name="GetData">
574         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/GetData" style="
                document"/>
575         <wsdl:input name="GetDataRequest">
576           <wsp:PolicyReference URI="#wsDualHttpBinding_GetData_Input_policy"/>
577           <soap12:body use="literal"/>
578         </wsdl:input>
579         <wsdl:output name="GetDataResponse">
580           <wsp:PolicyReference URI="#wsDualHttpBinding_GetData_output_policy"/>
581           <soap12:body use="literal"/>
582         </wsdl:output>
583       </wsdl:operation>
584       <wsdl:operation name="GetNewIds">
585         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/GetNewIds" style="
                document"/>
586         <wsdl:input name="GetNewIdsRequest">
587           <wsp:PolicyReference URI="#wsDualHttpBinding_GetNewIds_Input_policy"/>
588           <soap12:body use="literal"/>
589         </wsdl:input>
590         <wsdl:output name="GetNewIdsResponse">
591           <wsp:PolicyReference URI="#wsDualHttpBinding_GetNewIds_output_policy"/>
592           <soap12:body use="literal"/>
593         </wsdl:output>
594       </wsdl:operation>
595       <wsdl:operation name="AddUpdateData">
596         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/AddUpdateData"
                style="document"/>
597         <wsdl:input name="AddUpdateDataRequest">
598           <wsp:PolicyReference URI="#wsDualHttpBinding_AddUpdateData_Input_policy"/>
599           <soap12:body use="literal"/>
600         </wsdl:input>
601         <wsdl:output name="AddUpdateDataResponse">
602           <wsp:PolicyReference URI="#wsDualHttpBinding_AddUpdateData_output_policy"/>
603           <soap12:body use="literal"/>
604         </wsdl:output>
605       </wsdl:operation>
606       <wsdl:operation name="SubscribeLock">
607         <soap12:operation soapAction="http://tempuri.org/IDataStorageService/SubscribeLock"
                style="document"/>
608         <wsdl:input name="SubscribeLockRequest">
609           <wsp:PolicyReference URI="#wsDualHttpBinding_SubscribeLock_Input_policy"/>
610           <soap12:body use="literal"/>
611         </wsdl:input>
612         <wsdl:output name="SubscribeLockResponse">
613           <wsp:PolicyReference URI="#wsDualHttpBinding_SubscribeLock_output_policy"/>
614           <soap12:body use="literal"/>
```

```
615       </wsdl:output>
616     </wsdl:operation>
617     <wsdl:operation name="ReleaseLock">
618       <soap12:operation soapAction="http://tempuri.org/IDataStorageService/ReleaseLock" style=
              "document"/>
619       <wsdl:input name="ReleaseLockRequest">
620         <wsp:PolicyReference URI="#wsDualHttpBinding_ReleaseLock_Input_policy"/>
621         <soap12:body use="literal"/>
622       </wsdl:input>
623       <wsdl:output name="ReleaseLockResponse">
624         <wsp:PolicyReference URI="#wsDualHttpBinding_ReleaseLock_output_policy"/>
625         <soap12:body use="literal"/>
626       </wsdl:output>
627     </wsdl:operation>
628     <wsdl:operation name="NotifyLock">
629       <soap12:operation soapAction="http://tempuri.org/IDataStorageService/NotifyLock" style="
              document"/>
630       <wsdl:output name="NotifyLockNotification">
631         <wsp:PolicyReference URI="#wsDualHttpBinding_NotifyLock_output_policy"/>
632         <soap12:body use="literal"/>
633       </wsdl:output>
634     </wsdl:operation>
635   </wsdl:binding>
636   <wsdl:service name="DataStorageService">
637     <wsdl:port name="wsDualHttpBinding" binding="tns:wsDualHttpBinding">
638       <soap12:address location="http://localhost:8011/DataStorage"/>
639       <wsa10:EndpointReference>
640         <wsa10:Address>http://localhost:8011/DataStorage</wsa10:Address>
641         <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
642           <Upn>lucknerw@pdts.at</Upn>
643         </Identity>
644       </wsa10:EndpointReference>
645     </wsdl:port>
646   </wsdl:service>
647 </wsdl:definitions>
```

http://localhost:8011/?xsd=xsd0

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
       elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3    <xs:import schemaLocation="http://localhost:8011/?xsd=xsd2" namespace="http://schemas.
         microsoft.com/2003/10/Serialization/Arrays"/>
4    <xs:import schemaLocation="http://localhost:8011/?xsd=xsd3" namespace="http://schemas.
         datacontract.org/2004/07/DataStorage"/>
5    <xs:element name="AddNewUserReqeust">
6      <xs:complexType>
7        <xs:sequence>
8          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
9        </xs:sequence>
10     </xs:complexType>
11   </xs:element>
12   <xs:element name="AddNewUserResponse">
13     <xs:complexType>
14       <xs:sequence>
15         <xs:element name="Status" minOccurs="0" type="xs:int"/>
16         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
17       </xs:sequence>
18     </xs:complexType>
19   </xs:element>
20   <xs:element name="ListDataRequest">
21     <xs:complexType>
22       <xs:sequence>
23         <xs:element xmlns:q1="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
               "KeyIdList" minOccurs="0" nillable="true" type="q1:ArrayOfint"/>
24         <xs:element name="Type" minOccurs="0" nillable="true" type="xs:string"/>
25         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
26       </xs:sequence>
27     </xs:complexType>
28   </xs:element>
29   <xs:element name="ListDataResponse">
30     <xs:complexType>
31       <xs:sequence>
32         <xs:element xmlns:q2="http://schemas.datacontract.org/2004/07/DataStorage" name="
               DataSetInfoList" minOccurs="0" nillable="true" type="q2:ArrayOfDS_DataSetInfo"/>
33         <xs:element name="Status" minOccurs="0" type="xs:int"/>
34         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
35       </xs:sequence>
```

```
36      </xs:complexType>
37    </xs:element>
38    <xs:element name="GetDataRequest">
39      <xs:complexType>
40        <xs:sequence>
41          <xs:element xmlns:q3="http://schemas.datacontract.org/2004/07/DataStorage" name="
                IdKeyIdList" minOccurs="0" nillable="true" type="q3:ArrayOfDS_GetDataRequestItem"
                />
42          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
43        </xs:sequence>
44      </xs:complexType>
45    </xs:element>
46    <xs:element name="GetDataResponse">
47      <xs:complexType>
48        <xs:sequence>
49          <xs:element xmlns:q4="http://schemas.datacontract.org/2004/07/DataStorage" name="
                DatasetList" minOccurs="0" nillable="true" type="q4:ArrayOfDS_DataSet"/>
50          <xs:element name="Status" minOccurs="0" type="xs:int"/>
51          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
52        </xs:sequence>
53      </xs:complexType>
54    </xs:element>
55    <xs:element name="GetNewIdsRequest">
56      <xs:complexType>
57        <xs:sequence>
58          <xs:element name="Count" minOccurs="0" type="xs:int"/>
59          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
60        </xs:sequence>
61      </xs:complexType>
62    </xs:element>
63    <xs:element name="GetNewIdsResponse">
64      <xs:complexType>
65        <xs:sequence>
66          <xs:element xmlns:q5="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
                "IdList" minOccurs="0" nillable="true" type="q5:ArrayOfint"/>
67          <xs:element name="Status" minOccurs="0" type="xs:int"/>
68          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
69        </xs:sequence>
70      </xs:complexType>
71    </xs:element>
72    <xs:element name="AddUpdateDataRequest">
73      <xs:complexType>
74        <xs:sequence>
75          <xs:element name="LockString" minOccurs="0" nillable="true" type="xs:string"/>
76          <xs:element xmlns:q6="http://schemas.datacontract.org/2004/07/DataStorage" name="
                RequestList" minOccurs="0" nillable="true" type="
                q6:ArrayOfDS_AddUpdateDataRequest"/>
77          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
78        </xs:sequence>
79      </xs:complexType>
80    </xs:element>
81    <xs:element name="AddUpdateDataResponse">
82      <xs:complexType>
83        <xs:sequence>
84          <xs:element name="Status" minOccurs="0" type="xs:int"/>
85          <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
86        </xs:sequence>
87      </xs:complexType>
88    </xs:element>
89    <xs:element name="SubscribeLockRequest">
90      <xs:complexType>
91        <xs:sequence>
92          <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
93        </xs:sequence>
94      </xs:complexType>
95    </xs:element>
96    <xs:element name="SubscribeLockResponse">
97      <xs:complexType>
98        <xs:sequence>
99          <xs:element name="LockString" minOccurs="0" nillable="true" type="xs:string"/>
100         <xs:element name="Status" minOccurs="0" type="xs:int"/>
101         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
102       </xs:sequence>
103     </xs:complexType>
104   </xs:element>
105   <xs:element name="ReleaseLockRequest">
106     <xs:complexType>
107       <xs:sequence>
```

```
108         <xs:element name="LockString" minOccurs="0" nillable="true" type="xs:string"/>
109         <xs:element name="Username" minOccurs="0" nillable="true" type="xs:string"/>
110       </xs:sequence>
111     </xs:complexType>
112   </xs:element>
113   <xs:element name="ReleaseLockResponse">
114     <xs:complexType>
115       <xs:sequence>
116         <xs:element name="Status" minOccurs="0" type="xs:int"/>
117         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
118       </xs:sequence>
119     </xs:complexType>
120   </xs:element>
121   <xs:element name="NotifyLockNotification">
122     <xs:complexType>
123       <xs:sequence>
124         <xs:element name="Lockstring" minOccurs="0" nillable="true" type="xs:string"/>
125       </xs:sequence>
126     </xs:complexType>
127   </xs:element>
128 </xs:schema>
```

http://localhost:8011/?xsd=xsd1

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
       /2003/10/Serialization/" attributeFormDefault="qualified" elementFormDefault="qualified"
       targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/">
3    <xs:element name="anyType" nillable="true" type="xs:anyType"/>
4    <xs:element name="anyURI" nillable="true" type="xs:anyURI"/>
5    <xs:element name="base64Binary" nillable="true" type="xs:base64Binary"/>
6    <xs:element name="boolean" nillable="true" type="xs:boolean"/>
7    <xs:element name="byte" nillable="true" type="xs:byte"/>
8    <xs:element name="dateTime" nillable="true" type="xs:dateTime"/>
9    <xs:element name="decimal" nillable="true" type="xs:decimal"/>
10   <xs:element name="double" nillable="true" type="xs:double"/>
11   <xs:element name="float" nillable="true" type="xs:float"/>
12   <xs:element name="int" nillable="true" type="xs:int"/>
13   <xs:element name="long" nillable="true" type="xs:long"/>
14   <xs:element name="QName" nillable="true" type="xs:QName"/>
15   <xs:element name="short" nillable="true" type="xs:short"/>
16   <xs:element name="string" nillable="true" type="xs:string"/>
17   <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte"/>
18   <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt"/>
19   <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong"/>
20   <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort"/>
21   <xs:element name="char" nillable="true" type="tns:char"/>
22   <xs:simpleType name="char">
23     <xs:restriction base="xs:int"/>
24   </xs:simpleType>
25   <xs:element name="duration" nillable="true" type="tns:duration"/>
26   <xs:simpleType name="duration">
27     <xs:restriction base="xs:duration">
28       <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
29       <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
30       <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
31     </xs:restriction>
32   </xs:simpleType>
33   <xs:element name="guid" nillable="true" type="tns:guid"/>
34   <xs:simpleType name="guid">
35     <xs:restriction base="xs:string">
36       <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F
             ]{12}"/>
37     </xs:restriction>
38   </xs:simpleType>
39   <xs:attribute name="FactoryType" type="xs:QName"/>
40   <xs:attribute name="Id" type="xs:ID"/>
41   <xs:attribute name="Ref" type="xs:IDREF"/>
42 </xs:schema>
```

http://localhost:8011/?xsd=xsd2

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
       /2003/10/Serialization/Arrays" elementFormDefault="qualified" targetNamespace="http://
       schemas.microsoft.com/2003/10/Serialization/Arrays">
3    <xs:complexType name="ArrayOfint">
```

```
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="int" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfint" type="tns:ArrayOfint" nillable="true"/>
</xs:schema>
```

http://localhost:8011/?xsd=xsd3

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.datacontract.org/2004/07/DataStorage" elementFormDefault="qualified" targetNamespace="http://schemas.datacontract.org/2004/07/DataStorage">
  <xs:import schemaLocation="http://localhost:8011/?xsd=xsd2" namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"/>
  <xs:complexType name="ArrayOfDS_DataSetInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_DataSetInfo" nillable="true" type="tns:DS_DataSetInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfDS_DataSetInfo" nillable="true" type="tns:ArrayOfDS_DataSetInfo"/>
  <xs:complexType name="DS_DataSetInfo">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
      <xs:element name="Username" nillable="true" type="xs:string"/>
      <xs:element name="Type" nillable="true" type="xs:string"/>
      <xs:element xmlns:q1="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name="KeyIdList" nillable="true" type="q1:ArrayOfint"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="DS_DataSetInfo" nillable="true" type="tns:DS_DataSetInfo"/>
  <xs:complexType name="ArrayOfDS_GetDataRequestItem">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_GetDataRequestItem" nillable="true" type="tns:DS_GetDataRequestItem"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfDS_GetDataRequestItem" nillable="true" type="tns:ArrayOfDS_GetDataRequestItem"/>
  <xs:complexType name="DS_GetDataRequestItem">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
      <xs:element name="KeyID" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="DS_GetDataRequestItem" nillable="true" type="tns:DS_GetDataRequestItem"/>
  <xs:complexType name="ArrayOfDS_DataSet">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_DataSet" nillable="true" type="tns:DS_DataSet"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfDS_DataSet" nillable="true" type="tns:ArrayOfDS_DataSet"/>
  <xs:complexType name="DS_DataSet">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
      <xs:element name="Username" nillable="true" type="xs:string"/>
      <xs:element name="Type" nillable="true" type="xs:string"/>
      <xs:element name="EncryptedData" nillable="true" type="xs:base64Binary"/>
      <xs:element name="EncryptionAlgorithm" nillable="true" type="xs:string"/>
      <xs:element name="KeyList" nillable="true" type="tns:ArrayOfDS_KeyInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="DS_DataSet" nillable="true" type="tns:DS_DataSet"/>
  <xs:complexType name="ArrayOfDS_KeyInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_KeyInfo" nillable="true" type="tns:DS_KeyInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfDS_KeyInfo" nillable="true" type="tns:ArrayOfDS_KeyInfo"/>
  <xs:complexType name="DS_KeyInfo">
    <xs:sequence>
      <xs:element name="KeyID" type="xs:int"/>
      <xs:element name="SessionKey" nillable="true" type="xs:base64Binary"/>
      <xs:element name="EncryptionAlgorithm" nillable="true" type="xs:string"/>
    </xs:sequence>
```

```
61    </xs:complexType>
62    <xs:element name="DS_KeyInfo" nillable="true" type="tns:DS_KeyInfo"/>
63    <xs:complexType name="ArrayOfDS_AddUpdateDataRequest">
64      <xs:sequence>
65        <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_AddUpdateDataRequest" nillable=
              "true" type="tns:DS_AddUpdateDataRequest"/>
66      </xs:sequence>
67    </xs:complexType>
68    <xs:element name="ArrayOfDS_AddUpdateDataRequest" nillable="true" type="
          tns:ArrayOfDS_AddUpdateDataRequest"/>
69    <xs:complexType name="DS_AddUpdateDataRequest">
70      <xs:sequence>
71        <xs:element name="ID" type="xs:int"/>
72        <xs:element name="Operation" nillable="true" type="xs:string"/>
73        <xs:element name="Type" nillable="true" type="xs:string"/>
74        <xs:element name="EncryptedData" nillable="true" type="xs:base64Binary"/>
75        <xs:element name="EncryptionAlgorithm" nillable="true" type="xs:string"/>
76        <xs:element name="KeyList" nillable="true" type="tns:ArrayOfDS_KeyInfo"/>
77      </xs:sequence>
78    </xs:complexType>
79    <xs:element name="DS_AddUpdateDataRequest" nillable="true" type="tns:DS_AddUpdateDataRequest
          "/>
80  </xs:schema>
```

A.7.5 Interface automatischen Datenquellen und Datenspeicher

http://localhost:8014/?wsdl

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.
        xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
        wss-wssecurity-utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:tns="http://tempuri.org
        /" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas
        .xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/
        addressing/policy" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas
        .microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsaw="http://www.w3.org/2006/05/addressing
        /wsdl" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:wsa10="http://www.w3.
        org/2005/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" name="
        DataStorageForAutomaticMeterService" targetNamespace="http://tempuri.org/">
3     <wsp:Policy wsu:Id="wsDualHttpBinding_DataStorageForAutomaticMeter_policy">
4       <wsp:ExactlyOne>
5         <wsp:All>
6           <wsrm:RMAssertion xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy">
7             <wsrm:InactivityTimeout Milliseconds="600000"/>
8             <wsrm:AcknowledgementInterval Milliseconds="200"/>
9           </wsrm:RMAssertion>
10          <sp:SymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
11            <wsp:Policy>
12              <sp:ProtectionToken>
13                <wsp:Policy>
14                  <sp:SecureConversationToken sp:IncludeToken="http://schemas.xmlsoap.org/ws
                        /2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
15                    <wsp:Policy>
16                      <sp:RequireDerivedKeys/>
17                      <sp:BootstrapPolicy>
18                        <wsp:Policy>
19                          <sp:SignedParts>
20                            <sp:Body/>
21                            <sp:Header Name="ChannelInstance" Namespace="http://schemas.
                                  microsoft.com/ws/2005/02/duplex"/>
22                            <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
                                  "/>
23                            <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
24                            <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
25                            <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
26                            <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
27                            <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
28                            <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/
                                  addressing"/>
```

```
29                          </sp:SignedParts>
30                          <sp:EncryptedParts>
31                            <sp:Body/>
32                          </sp:EncryptedParts>
33                          <sp:SymmetricBinding>
34                            <wsp:Policy>
35                              <sp:ProtectionToken>
36                                <wsp:Policy>
37                                  <sp:SpnegoContextToken sp:IncludeToken="http://schemas.xmlsoap
                                        .org/ws/2005/07/securitypolicy/IncludeToken/
                                        AlwaysToRecipient">
38                                    <wsp:Policy>
39                                      <sp:RequireDerivedKeys/>
40                                    </wsp:Policy>
41                                  </sp:SpnegoContextToken>
42                                </wsp:Policy>
43                              </sp:ProtectionToken>
44                              <sp:AlgorithmSuite>
45                                <wsp:Policy>
46                                  <sp:Basic256/>
47                                </wsp:Policy>
48                              </sp:AlgorithmSuite>
49                              <sp:Layout>
50                                <wsp:Policy>
51                                  <sp:Strict/>
52                                </wsp:Policy>
53                              </sp:Layout>
54                              <sp:IncludeTimestamp/>
55                              <sp:EncryptSignature/>
56                              <sp:OnlySignEntireHeadersAndBody/>
57                            </wsp:Policy>
58                          </sp:SymmetricBinding>
59                          <sp:Wss11>
60                            <wsp:Policy>
61                              <sp:MustSupportRefKeyIdentifier/>
62                              <sp:MustSupportRefIssuerSerial/>
63                              <sp:MustSupportRefThumbprint/>
64                              <sp:MustSupportRefEncryptedKey/>
65                            </wsp:Policy>
66                          </sp:Wss11>
67                          <sp:Trust10>
68                            <wsp:Policy>
69                              <sp:MustSupportIssuedTokens/>
70                              <sp:RequireClientEntropy/>
71                              <sp:RequireServerEntropy/>
72                            </wsp:Policy>
73                          </sp:Trust10>
74                        </wsp:Policy>
75                      </sp:BootstrapPolicy>
76                    </wsp:Policy>
77                  </sp:SecureConversationToken>
78                </wsp:Policy>
79              </sp:ProtectionToken>
80              <sp:AlgorithmSuite>
81                <wsp:Policy>
82                  <sp:Basic256/>
83                </wsp:Policy>
84              </sp:AlgorithmSuite>
85              <sp:Layout>
86                <wsp:Policy>
87                  <sp:Strict/>
88                </wsp:Policy>
89              </sp:Layout>
90              <sp:IncludeTimestamp/>
91              <sp:EncryptSignature/>
92              <sp:OnlySignEntireHeadersAndBody/>
93            </wsp:Policy>
94          </sp:SymmetricBinding>
95          <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
96            <wsp:Policy>
97              <sp:MustSupportRefKeyIdentifier/>
98              <sp:MustSupportRefIssuerSerial/>
99              <sp:MustSupportRefThumbprint/>
100             <sp:MustSupportRefEncryptedKey/>
101           </wsp:Policy>
102         </sp:Wss11>
103         <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
104           <wsp:Policy>
```

```
105             <sp:MustSupportIssuedTokens/>
106             <sp:RequireClientEntropy/>
107             <sp:RequireServerEntropy/>
108           </wsp:Policy>
109         </sp:Trust10>
110         <cdp:CompositeDuplex xmlns:cdp="http://schemas.microsoft.com/net/2006/06/duplex"/>
111         <ow:OneWay xmlns:ow="http://schemas.microsoft.com/ws/2005/05/routing/policy"/>
112         <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/
                policy/optimizedmimeserialization"/>
113         <wsaw:UsingAddressing/>
114       </wsp:All>
115     </wsp:ExactlyOne>
116   </wsp:Policy>
117   <wsp:Policy wsu:Id="
          wsDualHttpBinding_DataStorageForAutomaticMeter_GetDataForAutomaticMeter_Input_policy">
118     <wsp:ExactlyOne>
119       <wsp:All>
120         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
121           <sp:Body/>
122           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
123           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
124           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
125           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
126           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
127           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
128           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
129           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
130           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
131           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
132           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
133         </sp:SignedParts>
134         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
135           <sp:Body/>
136         </sp:EncryptedParts>
137       </wsp:All>
138     </wsp:ExactlyOne>
139   </wsp:Policy>
140   <wsp:Policy wsu:Id="
          wsDualHttpBinding_DataStorageForAutomaticMeter_GetDataForAutomaticMeter_output_policy">
141     <wsp:ExactlyOne>
142       <wsp:All>
143         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
144           <sp:Body/>
145           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
146           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
147           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
148           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
149           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
150           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
151           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
152           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
153           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
154           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
155           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
156         </sp:SignedParts>
157         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
158           <sp:Body/>
159         </sp:EncryptedParts>
160       </wsp:All>
161     </wsp:ExactlyOne>
162   </wsp:Policy>
163   <wsp:Policy wsu:Id="
          wsDualHttpBinding_DataStorageForAutomaticMeter_GetNewIdsForAutomaticMeter_Input_policy"
          >
164     <wsp:ExactlyOne>
165       <wsp:All>
166         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
167           <sp:Body/>
168           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
169           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
```

```
170             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
171             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
172             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
173             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
174             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
175             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
176             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
177             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
178             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
179           </sp:SignedParts>
180           <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
181             <sp:Body/>
182           </sp:EncryptedParts>
183         </wsp:All>
184       </wsp:ExactlyOne>
185     </wsp:Policy>
186     <wsp:Policy wsu:Id="
            wsDualHttpBinding_DataStorageForAutomaticMeter_GetNewIdsForAutomaticMeter_output_policy
            ">
187       <wsp:ExactlyOne>
188         <wsp:All>
189           <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
190             <sp:Body/>
191             <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
192             <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                    /2005/02/rm"/>
193             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
194             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
195             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
196             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
197             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
198             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
199             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
200             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
201             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
202           </sp:SignedParts>
203           <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
204             <sp:Body/>
205           </sp:EncryptedParts>
206         </wsp:All>
207       </wsp:ExactlyOne>
208     </wsp:Policy>
209     <wsp:Policy wsu:Id="
            wsDualHttpBinding_DataStorageForAutomaticMeter_AddUpdateDataForAutomaticMeter_Input_policy
            ">
210       <wsp:ExactlyOne>
211         <wsp:All>
212           <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
213             <sp:Body/>
214             <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
215             <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                    /2005/02/rm"/>
216             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
217             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
218             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
219             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
220             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
221             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
222             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
223             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
224             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
225           </sp:SignedParts>
226           <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
227             <sp:Body/>
228           </sp:EncryptedParts>
229         </wsp:All>
230       </wsp:ExactlyOne>
231     </wsp:Policy>
232     <wsp:Policy wsu:Id="
            wsDualHttpBinding_DataStorageForAutomaticMeter_AddUpdateDataForAutomaticMeter_output_policy
            ">
233       <wsp:ExactlyOne>
```

```
234       <wsp:All>
235         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
236           <sp:Body/>
237           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
238           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
239           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
240           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
241           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
242           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
243           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
244           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
245           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
246           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
247           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
248         </sp:SignedParts>
249         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
250           <sp:Body/>
251         </sp:EncryptedParts>
252       </wsp:All>
253     </wsp:ExactlyOne>
254   </wsp:Policy>
255   <wsp:Policy wsu:Id="
          wsDualHttpBinding_DataStorageForAutomaticMeter_SubscribeLockForAutomaticmeter_Input_policy
          ">
256     <wsp:ExactlyOne>
257       <wsp:All>
258         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
259           <sp:Body/>
260           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
261           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
262           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
263           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
264           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
265           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
266           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
267           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
268           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
269           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
270           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
271         </sp:SignedParts>
272         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
273           <sp:Body/>
274         </sp:EncryptedParts>
275       </wsp:All>
276     </wsp:ExactlyOne>
277   </wsp:Policy>
278   <wsp:Policy wsu:Id="
          wsDualHttpBinding_DataStorageForAutomaticMeter_SubscribeLockForAutomaticmeter_output_policy
          ">
279     <wsp:ExactlyOne>
280       <wsp:All>
281         <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
282           <sp:Body/>
283           <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
284           <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                /2005/02/rm"/>
285           <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                >
286           <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                /2005/02/duplex"/>
287           <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
288           <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
289           <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
290           <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
291           <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
292           <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
293           <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
294         </sp:SignedParts>
295         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
296           <sp:Body/>
297         </sp:EncryptedParts>
298       </wsp:All>
```

```
299       </wsp:ExactlyOne>
300     </wsp:Policy>
301     <wsp:Policy wsu:Id="
            wsDualHttpBinding_DataStorageForAutomaticMeter_ReleaseLockForAutomaticmeter_Input_policy
            ">
302       <wsp:ExactlyOne>
303         <wsp:All>
304           <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
305             <sp:Body/>
306             <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
307             <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                    /2005/02/rm"/>
308             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
309             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
310             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
311             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
312             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
313             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
314             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
315             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
316             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
317           </sp:SignedParts>
318           <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
319             <sp:Body/>
320           </sp:EncryptedParts>
321         </wsp:All>
322       </wsp:ExactlyOne>
323     </wsp:Policy>
324     <wsp:Policy wsu:Id="
            wsDualHttpBinding_DataStorageForAutomaticMeter_ReleaseLockForAutomaticmeter_output_policy
            ">
325       <wsp:ExactlyOne>
326         <wsp:All>
327           <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
328             <sp:Body/>
329             <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
330             <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                    /2005/02/rm"/>
331             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
332             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
333             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
334             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
335             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
336             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
337             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
338             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
339             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
340           </sp:SignedParts>
341           <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
342             <sp:Body/>
343           </sp:EncryptedParts>
344         </wsp:All>
345       </wsp:ExactlyOne>
346     </wsp:Policy>
347     <wsp:Policy wsu:Id="wsDualHttpBinding_DataStorageForAutomaticMeter_NotifyLock_output_policy"
            >
348       <wsp:ExactlyOne>
349         <wsp:All>
350           <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
351             <sp:Body/>
352             <sp:Header Name="Sequence" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
353             <sp:Header Name="SequenceAcknowledgement" Namespace="http://schemas.xmlsoap.org/ws
                    /2005/02/rm"/>
354             <sp:Header Name="AckRequested" Namespace="http://schemas.xmlsoap.org/ws/2005/02/rm"/
                    >
355             <sp:Header Name="ChannelInstance" Namespace="http://schemas.microsoft.com/ws
                    /2005/02/duplex"/>
356             <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
357             <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
358             <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
359             <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
360             <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
361             <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
362             <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
```

```
363         </sp:SignedParts>
364         <sp:EncryptedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
365           <sp:Body/>
366         </sp:EncryptedParts>
367       </wsp:All>
368     </wsp:ExactlyOne>
369   </wsp:Policy>
370   <wsdl:types>
371     <xsd:schema targetNamespace="http://tempuri.org/Imports">
372       <xsd:import schemaLocation="http://localhost:8014/?xsd=xsd0" namespace="http://tempuri.
              org/"/>
373       <xsd:import schemaLocation="http://localhost:8014/?xsd=xsd1" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/"/>
374       <xsd:import schemaLocation="http://localhost:8014/?xsd=xsd2" namespace="http://schemas.
              datacontract.org/2004/07/DataStorage"/>
375       <xsd:import schemaLocation="http://localhost:8014/?xsd=xsd3" namespace="http://schemas.
              microsoft.com/2003/10/Serialization/Arrays"/>
376     </xsd:schema>
377   </wsdl:types>
378   <wsdl:message name="GetDataForAutomaticmeterRequest">
379     <wsdl:part name="parameters" element="tns:GetDataForAutomaticmeterRequest"/>
380   </wsdl:message>
381   <wsdl:message name="GetDataForAutomaticmeterResponse">
382     <wsdl:part name="parameters" element="tns:GetDataForAutomaticmeterResponse"/>
383   </wsdl:message>
384   <wsdl:message name="GetNewIdsForAutomaticmeterRequest">
385     <wsdl:part name="parameters" element="tns:GetNewIdsForAutomaticmeterRequest"/>
386   </wsdl:message>
387   <wsdl:message name="GetNewIdsForAutomaticmeterResponse">
388     <wsdl:part name="parameters" element="tns:GetNewIdsForAutomaticmeterResponse"/>
389   </wsdl:message>
390   <wsdl:message name="AddUpdateDataForAutomaticmeterRequest">
391     <wsdl:part name="parameters" element="tns:AddUpdateDataForAutomaticmeterRequest"/>
392   </wsdl:message>
393   <wsdl:message name="AddUpdateDataForAutomaticmeterResponse">
394     <wsdl:part name="parameters" element="tns:AddUpdateDataForAutomaticmeterResponse"/>
395   </wsdl:message>
396   <wsdl:message name="SubscribeLockForAutomaticmeterRequest">
397     <wsdl:part name="parameters" element="tns:SubscribeLockForAutomaticmeterRequest"/>
398   </wsdl:message>
399   <wsdl:message name="SubscribeLockForAutomaticmeterResponse">
400     <wsdl:part name="parameters" element="tns:SubscribeLockForAutomaticmeterResponse"/>
401   </wsdl:message>
402   <wsdl:message name="ReleaseLockForAutomaticmeterRequest">
403     <wsdl:part name="parameters" element="tns:ReleaseLockForAutomaticmeterRequest"/>
404   </wsdl:message>
405   <wsdl:message name="ReleaseLockForAutomaticmeterResponse">
406     <wsdl:part name="parameters" element="tns:ReleaseLockForAutomaticmeterResponse"/>
407   </wsdl:message>
408   <wsdl:message name="NotifyLockNotification">
409     <wsdl:part name="parameters" element="tns:NotifyLockNotification"/>
410   </wsdl:message>
411   <wsdl:portType name="IDataStorageForAutomaticMeterService">
412     <wsdl:operation name="GetDataForAutomaticMeter">
413       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetDataForAutomaticMeter" name="GetDataForAutomaticmeterRequest" message="
              tns:GetDataForAutomaticmeterRequest"/>
414       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetDataForAutomaticMeterResponse" name="GetDataForAutomaticmeterResponse" message="
              tns:GetDataForAutomaticmeterResponse"/>
415     </wsdl:operation>
416     <wsdl:operation name="GetNewIdsForAutomaticMeter">
417       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetNewIdsForAutomaticMeter" name="GetNewIdsForAutomaticmeterRequest" message="
              tns:GetNewIdsForAutomaticmeterRequest"/>
418       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetNewIdsForAutomaticMeterResponse" name="GetNewIdsForAutomaticmeterResponse"
              message="tns:GetNewIdsForAutomaticmeterResponse"/>
419     </wsdl:operation>
420     <wsdl:operation name="AddUpdateDataForAutomaticMeter">
421       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              AddUpdateDataForAutomaticMeter" name="AddUpdateDataForAutomaticmeterRequest"
              message="tns:AddUpdateDataForAutomaticmeterRequest"/>
422       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              AddUpdateDataForAutomaticMeterResponse" name="
              AddUpdateDataForAutomaticmeterResponse" message="
              tns:AddUpdateDataForAutomaticmeterResponse"/>
423     </wsdl:operation>
```

```
424     <wsdl:operation name="SubscribeLockForAutomaticmeter">
425       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              SubscribeLockForAutomaticmeter" name="SubscribeLockForAutomaticmeterRequest"
              message="tns:SubscribeLockForAutomaticmeterRequest"/>
426       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              SubscribeLockForAutomaticmeterResponse" name="
              SubscribeLockForAutomaticmeterResponse" message="
              tns:SubscribeLockForAutomaticmeterResponse"/>
427     </wsdl:operation>
428     <wsdl:operation name="ReleaseLockForAutomaticmeter">
429       <wsdl:input wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              ReleaseLockForAutomaticmeter" name="ReleaseLockForAutomaticmeterRequest" message="
              tns:ReleaseLockForAutomaticmeterRequest"/>
430       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              ReleaseLockForAutomaticmeterResponse" name="ReleaseLockForAutomaticmeterResponse"
              message="tns:ReleaseLockForAutomaticmeterResponse"/>
431     </wsdl:operation>
432     <wsdl:operation name="NotifyLock">
433       <wsdl:output wsaw:Action="http://tempuri.org/IDataStorageForAutomaticMeterService/
              NotifyLock" name="NotifyLockNotification" message="tns:NotifyLockNotification"/>
434     </wsdl:operation>
435   </wsdl:portType>
436   <wsdl:binding name="wsDualHttpBinding_DataStorageForAutomaticMeter" type="
          tns:IDataStorageForAutomaticMeterService">
437     <wsp:PolicyReference URI="#wsDualHttpBinding_DataStorageForAutomaticMeter_policy"/>
438     <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
439     <wsdl:operation name="GetDataForAutomaticMeter">
440       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetDataForAutomaticMeter" style="document"/>
441       <wsdl:input name="GetDataForAutomaticmeterRequest">
442         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_GetDataForAutomaticMeter_Input_policy
                "/>
443         <soap12:body use="literal"/>
444       </wsdl:input>
445       <wsdl:output name="GetDataForAutomaticmeterResponse">
446         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_GetDataForAutomaticMeter_output_policy
                "/>
447         <soap12:body use="literal"/>
448       </wsdl:output>
449     </wsdl:operation>
450     <wsdl:operation name="GetNewIdsForAutomaticMeter">
451       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              GetNewIdsForAutomaticMeter" style="document"/>
452       <wsdl:input name="GetNewIdsForAutomaticmeterRequest">
453         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_GetNewIdsForAutomaticMeter_Input_policy
                "/>
454         <soap12:body use="literal"/>
455       </wsdl:input>
456       <wsdl:output name="GetNewIdsForAutomaticmeterResponse">
457         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_GetNewIdsForAutomaticMeter_output_policy
                "/>
458         <soap12:body use="literal"/>
459       </wsdl:output>
460     </wsdl:operation>
461     <wsdl:operation name="AddUpdateDataForAutomaticMeter">
462       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              AddUpdateDataForAutomaticMeter" style="document"/>
463       <wsdl:input name="AddUpdateDataForAutomaticmeterRequest">
464         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_AddUpdateDataForAutomaticMeter_Input_policy
                "/>
465         <soap12:body use="literal"/>
466       </wsdl:input>
467       <wsdl:output name="AddUpdateDataForAutomaticmeterResponse">
468         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_AddUpdateDataForAutomaticMeter_output_policy
                "/>
469         <soap12:body use="literal"/>
470       </wsdl:output>
471     </wsdl:operation>
472     <wsdl:operation name="SubscribeLockForAutomaticmeter">
473       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              SubscribeLockForAutomaticmeter" style="document"/>
474       <wsdl:input name="SubscribeLockForAutomaticmeterRequest">
```

```
475         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_SubscribeLockForAutomaticmeter_Input_policy
                "/>
476         <soap12:body use="literal"/>
477       </wsdl:input>
478       <wsdl:output name="SubscribeLockForAutomaticmeterResponse">
479         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_SubscribeLockForAutomaticmeter_output_policy
                "/>
480         <soap12:body use="literal"/>
481       </wsdl:output>
482     </wsdl:operation>
483     <wsdl:operation name="ReleaseLockForAutomaticmeter">
484       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              ReleaseLockForAutomaticmeter" style="document"/>
485       <wsdl:input name="ReleaseLockForAutomaticmeterRequest">
486         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_ReleaseLockForAutomaticmeter_Input_policy
                "/>
487         <soap12:body use="literal"/>
488       </wsdl:input>
489       <wsdl:output name="ReleaseLockForAutomaticmeterResponse">
490         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_ReleaseLockForAutomaticmeter_output_policy
                "/>
491         <soap12:body use="literal"/>
492       </wsdl:output>
493     </wsdl:operation>
494     <wsdl:operation name="NotifyLock">
495       <soap12:operation soapAction="http://tempuri.org/IDataStorageForAutomaticMeterService/
              NotifyLock" style="document"/>
496       <wsdl:output name="NotifyLockNotification">
497         <wsp:PolicyReference URI="#
                wsDualHttpBinding_DataStorageForAutomaticMeter_NotifyLock_output_policy"/>
498         <soap12:body use="literal"/>
499       </wsdl:output>
500     </wsdl:operation>
501   </wsdl:binding>
502   <wsdl:service name="DataStorageForAutomaticMeterService">
503     <wsdl:port name="wsDualHttpBinding_DataStorageForAutomaticMeter" binding="
            tns:wsDualHttpBinding_DataStorageForAutomaticMeter">
504       <soap12:address location="http://localhost:8014/DataStorageForAutomaticMeter"/>
505       <wsa10:EndpointReference>
506         <wsa10:Address>http://localhost:8014/DataStorageForAutomaticMeter</wsa10:Address>
507         <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
508           <Upn>lucknerw@pdts.at</Upn>
509         </Identity>
510       </wsa10:EndpointReference>
511     </wsdl:port>
512   </wsdl:service>
513 </wsdl:definitions>
```

http://localhost:8014/?xsd=xsd0

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
       elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3    <xs:import schemaLocation="http://localhost:8014/?xsd=xsd2" namespace="http://schemas.
         datacontract.org/2004/07/DataStorage"/>
4    <xs:import schemaLocation="http://localhost:8014/?xsd=xsd3" namespace="http://schemas.
         microsoft.com/2003/10/Serialization/Arrays"/>
5    <xs:element name="GetDataForAutomaticmeterRequest">
6      <xs:complexType>
7        <xs:sequence>
8          <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
9          <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
10         <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
11         <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
12       </xs:sequence>
13     </xs:complexType>
14   </xs:element>
15   <xs:element name="GetDataForAutomaticmeterResponse">
16     <xs:complexType>
17       <xs:sequence>
18         <xs:element xmlns:q1="http://schemas.datacontract.org/2004/07/DataStorage" name="
               DatasetList" minOccurs="0" nillable="true" type="q1:ArrayOfDS_DataSet"/>
19         <xs:element name="Status" minOccurs="0" type="xs:int"/>
20         <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
```

```
21      </xs:sequence>
22    </xs:complexType>
23  </xs:element>
24  <xs:element name="GetNewIdsForAutomaticmeterRequest">
25    <xs:complexType>
26      <xs:sequence>
27        <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
28        <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
29        <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
30        <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
31      </xs:sequence>
32    </xs:complexType>
33  </xs:element>
34  <xs:element name="GetNewIdsForAutomaticmeterResponse">
35    <xs:complexType>
36      <xs:sequence>
37        <xs:element xmlns:q2="http://schemas.microsoft.com/2003/10/Serialization/Arrays" name=
              "IdList" minOccurs="0" nillable="true" type="q2:ArrayOfint"/>
38        <xs:element name="Status" minOccurs="0" type="xs:int"/>
39        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
40      </xs:sequence>
41    </xs:complexType>
42  </xs:element>
43  <xs:element name="AddUpdateDataForAutomaticmeterRequest">
44    <xs:complexType>
45      <xs:sequence>
46        <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
47        <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
48        <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
49        <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
50      </xs:sequence>
51    </xs:complexType>
52  </xs:element>
53  <xs:element name="AddUpdateDataForAutomaticmeterResponse">
54    <xs:complexType>
55      <xs:sequence>
56        <xs:element name="Status" minOccurs="0" type="xs:int"/>
57        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
58      </xs:sequence>
59    </xs:complexType>
60  </xs:element>
61  <xs:element name="SubscribeLockForAutomaticmeterRequest">
62    <xs:complexType>
63      <xs:sequence>
64        <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
65        <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
66        <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
67        <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
68      </xs:sequence>
69    </xs:complexType>
70  </xs:element>
71  <xs:element name="SubscribeLockForAutomaticmeterResponse">
72    <xs:complexType>
73      <xs:sequence>
74        <xs:element name="LockString" minOccurs="0" nillable="true" type="xs:string"/>
75        <xs:element name="Status" minOccurs="0" type="xs:int"/>
76        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
77      </xs:sequence>
78    </xs:complexType>
79  </xs:element>
80  <xs:element name="ReleaseLockForAutomaticmeterRequest">
81    <xs:complexType>
82      <xs:sequence>
83        <xs:element name="Signature" minOccurs="0" nillable="true" type="xs:string"/>
84        <xs:element name="SignatureAlgorithm" minOccurs="0" nillable="true" type="xs:string"/>
85        <xs:element name="SignatureKeyId" minOccurs="0" type="xs:int"/>
86        <xs:element name="XmlRequest" minOccurs="0" nillable="true" type="xs:string"/>
87      </xs:sequence>
88    </xs:complexType>
89  </xs:element>
90  <xs:element name="ReleaseLockForAutomaticmeterResponse">
91    <xs:complexType>
92      <xs:sequence>
93        <xs:element name="Status" minOccurs="0" type="xs:int"/>
94        <xs:element name="StatusMessage" minOccurs="0" nillable="true" type="xs:string"/>
95      </xs:sequence>
96    </xs:complexType>
97  </xs:element>
```

```
98    <xs:element name="NotifyLockNotification">
99      <xs:complexType>
100       <xs:sequence>
101         <xs:element name="Lockstring" minOccurs="0" nillable="true" type="xs:string"/>
102       </xs:sequence>
103     </xs:complexType>
104   </xs:element>
105 </xs:schema>
```

http://localhost:8014/?xsd=xsd1

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
        /2003/10/Serialization/" attributeFormDefault="qualified" elementFormDefault="qualified"
        targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/">
3     <xs:element name="anyType" nillable="true" type="xs:anyType"/>
4     <xs:element name="anyURI" nillable="true" type="xs:anyURI"/>
5     <xs:element name="base64Binary" nillable="true" type="xs:base64Binary"/>
6     <xs:element name="boolean" nillable="true" type="xs:boolean"/>
7     <xs:element name="byte" nillable="true" type="xs:byte"/>
8     <xs:element name="dateTime" nillable="true" type="xs:dateTime"/>
9     <xs:element name="decimal" nillable="true" type="xs:decimal"/>
10    <xs:element name="double" nillable="true" type="xs:double"/>
11    <xs:element name="float" nillable="true" type="xs:float"/>
12    <xs:element name="int" nillable="true" type="xs:int"/>
13    <xs:element name="long" nillable="true" type="xs:long"/>
14    <xs:element name="QName" nillable="true" type="xs:QName"/>
15    <xs:element name="short" nillable="true" type="xs:short"/>
16    <xs:element name="string" nillable="true" type="xs:string"/>
17    <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte"/>
18    <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt"/>
19    <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong"/>
20    <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort"/>
21    <xs:element name="char" nillable="true" type="tns:char"/>
22    <xs:simpleType name="char">
23      <xs:restriction base="xs:int"/>
24    </xs:simpleType>
25    <xs:element name="duration" nillable="true" type="tns:duration"/>
26    <xs:simpleType name="duration">
27      <xs:restriction base="xs:duration">
28        <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?"/>
29        <xs:minInclusive value="-P10675199DT2H48M5.4775808S"/>
30        <xs:maxInclusive value="P10675199DT2H48M5.4775807S"/>
31      </xs:restriction>
32    </xs:simpleType>
33    <xs:element name="guid" nillable="true" type="tns:guid"/>
34    <xs:simpleType name="guid">
35      <xs:restriction base="xs:string">
36        <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F
              ]{12}"/>
37      </xs:restriction>
38    </xs:simpleType>
39    <xs:attribute name="FactoryType" type="xs:QName"/>
40    <xs:attribute name="Id" type="xs:ID"/>
41    <xs:attribute name="Ref" type="xs:IDREF"/>
42  </xs:schema>
```

http://localhost:8014/?xsd=xsd2

```
1   <?xml version='1.0' encoding='UTF-8'?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.datacontract.
        org/2004/07/DataStorage" elementFormDefault="qualified" targetNamespace="http://schemas.
        datacontract.org/2004/07/DataStorage">
3     <xs:complexType name="ArrayOfDS_DataSet">
4       <xs:sequence>
5         <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_DataSet" nillable="true" type="
              tns:DS_DataSet"/>
6       </xs:sequence>
7     </xs:complexType>
8     <xs:element name="ArrayOfDS_DataSet" nillable="true" type="tns:ArrayOfDS_DataSet"/>
9     <xs:complexType name="DS_DataSet">
10      <xs:sequence>
11        <xs:element name="ID" type="xs:int"/>
12        <xs:element name="Username" nillable="true" type="xs:string"/>
13        <xs:element name="Type" nillable="true" type="xs:string"/>
14        <xs:element name="EncryptedData" nillable="true" type="xs:base64Binary"/>
15        <xs:element name="EncryptionAlgorithm" nillable="true" type="xs:string"/>
```

```
16      <xs:element name="KeyList" nillable="true" type="tns:ArrayOfDS_KeyInfo"/>
17    </xs:sequence>
18  </xs:complexType>
19  <xs:element name="DS_DataSet" nillable="true" type="tns:DS_DataSet"/>
20  <xs:complexType name="ArrayOfDS_KeyInfo">
21    <xs:sequence>
22      <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_KeyInfo" nillable="true" type="
            tns:DS_KeyInfo"/>
23    </xs:sequence>
24  </xs:complexType>
25  <xs:element name="ArrayOfDS_KeyInfo" nillable="true" type="tns:ArrayOfDS_KeyInfo"/>
26  <xs:complexType name="DS_KeyInfo">
27    <xs:sequence>
28      <xs:element name="KeyID" type="xs:int"/>
29      <xs:element name="SessionKey" nillable="true" type="xs:base64Binary"/>
30      <xs:element name="EncryptionAlgorithm" nillable="true" type="xs:string"/>
31    </xs:sequence>
32  </xs:complexType>
33  <xs:element name="DS_KeyInfo" nillable="true" type="tns:DS_KeyInfo"/>
34 </xs:schema>
```

http://localhost:8014/?xsd=xsd3

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com
       /2003/10/Serialization/Arrays" elementFormDefault="qualified" targetNamespace="http://
       schemas.microsoft.com/2003/10/Serialization/Arrays">
3    <xs:complexType name="ArrayOfint">
4      <xs:sequence>
5        <xs:element minOccurs="0" maxOccurs="unbounded" name="int" type="xs:int"/>
6      </xs:sequence>
7    </xs:complexType>
8    <xs:element name="ArrayOfint" type="tns:ArrayOfint" nillable="true"/>
9  </xs:schema>
```

AddUpdateDataXmlRequest

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="AddUpdateDataXmlRequest" nillable="true" type="AddUpdateDataXmlRequest" />
4    <xs:complexType name="AddUpdateDataXmlRequest">
5      <xs:complexContent mixed="false">
6        <xs:extension base="AddUpdateDataRequest" />
7      </xs:complexContent>
8    </xs:complexType>
9    <xs:complexType name="AddUpdateDataRequest">
10     <xs:sequence>
11       <xs:element minOccurs="0" maxOccurs="1" name="RequestList" type="
             ArrayOfDS_AddUpdateDataRequest" />
12       <xs:element minOccurs="0" maxOccurs="1" name="LockString" type="xs:string" />
13       <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
14     </xs:sequence>
15   </xs:complexType>
16   <xs:complexType name="ArrayOfDS_AddUpdateDataRequest">
17     <xs:sequence>
18       <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_AddUpdateDataRequest" nillable=
             "true" type="DS_AddUpdateDataRequest" />
19     </xs:sequence>
20   </xs:complexType>
21   <xs:complexType name="DS_AddUpdateDataRequest">
22     <xs:sequence>
23       <xs:element minOccurs="0" maxOccurs="1" name="Operation" type="xs:string" />
24       <xs:element minOccurs="0" maxOccurs="1" name="EncryptionAlgorithm" type="xs:string" />
25       <xs:element minOccurs="1" maxOccurs="1" name="Id" type="xs:int" />
26       <xs:element minOccurs="0" maxOccurs="1" name="Type" type="xs:string" />
27       <xs:element minOccurs="0" maxOccurs="1" name="EncryptedData" type="xs:base64Binary" />
28       <xs:element minOccurs="0" maxOccurs="1" name="KeyList" type="ArrayOfDS_KeyInfo" />
29     </xs:sequence>
30   </xs:complexType>
31   <xs:complexType name="ArrayOfDS_KeyInfo">
32     <xs:sequence>
33       <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_KeyInfo" nillable="true" type="
             DS_KeyInfo" />
34     </xs:sequence>
35   </xs:complexType>
36   <xs:complexType name="DS_KeyInfo">
37     <xs:sequence>
```

```
38        <xs:element minOccurs="1" maxOccurs="1" name="KeyID" type="xs:int" />
39        <xs:element minOccurs="0" maxOccurs="1" name="SessionKey" type="xs:base64Binary" />
40        <xs:element minOccurs="0" maxOccurs="1" name="EncryptionAlgorithm" type="xs:string" />
41      </xs:sequence>
42    </xs:complexType>
43  </xs:schema>
```

GetDataXmlRequest

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3     <xs:element name="GetDataXmlRequest" nillable="true" type="GetDataXmlRequest" />
4     <xs:complexType name="GetDataXmlRequest">
5       <xs:complexContent mixed="false">
6         <xs:extension base="GetDataRequest" />
7       </xs:complexContent>
8     </xs:complexType>
9     <xs:complexType name="GetDataRequest">
10      <xs:sequence>
11        <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
12        <xs:element minOccurs="0" maxOccurs="1" name="IdKeyIdList" type="
              ArrayOfDS_GetDataRequestItem" />
13      </xs:sequence>
14    </xs:complexType>
15    <xs:complexType name="ArrayOfDS_GetDataRequestItem">
16      <xs:sequence>
17        <xs:element minOccurs="0" maxOccurs="unbounded" name="DS_GetDataRequestItem" nillable="
              true" type="DS_GetDataRequestItem" />
18      </xs:sequence>
19    </xs:complexType>
20    <xs:complexType name="DS_GetDataRequestItem">
21      <xs:sequence>
22        <xs:element minOccurs="1" maxOccurs="1" name="ID" type="xs:int" />
23        <xs:element minOccurs="1" maxOccurs="1" name="KeyId" type="xs:int" />
24      </xs:sequence>
25    </xs:complexType>
26  </xs:schema>
```

GetNewIdsXmlRequest

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3     <xs:element name="GetNewIdsXmlRequest" nillable="true" type="GetNewIdsXmlRequest" />
4     <xs:complexType name="GetNewIdsXmlRequest">
5       <xs:complexContent mixed="false">
6         <xs:extension base="GetNewIdsRequest" />
7       </xs:complexContent>
8     </xs:complexType>
9     <xs:complexType name="GetNewIdsRequest">
10      <xs:sequence>
11        <xs:element minOccurs="1" maxOccurs="1" name="Count" type="xs:int" />
12        <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
13      </xs:sequence>
14    </xs:complexType>
15  </xs:schema>
```

SubscribeLockXmlRequest

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3     <xs:element name="SubscribeLockXmlRequest" nillable="true" type="SubscribeLockXmlRequest" />
4     <xs:complexType name="SubscribeLockXmlRequest">
5       <xs:complexContent mixed="false">
6         <xs:extension base="SubscribeLockRequest" />
7       </xs:complexContent>
8     </xs:complexType>
9     <xs:complexType name="SubscribeLockRequest">
10      <xs:sequence>
11        <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
12      </xs:sequence>
13    </xs:complexType>
14  </xs:schema>
```

ReleaseLockXmlRequest

```
1   <?xml version="1.0" encoding="utf-8"?>
```

```
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ReleaseLockXmlRequest" nillable="true" type="ReleaseLockXmlRequest" />
  <xs:complexType name="ReleaseLockXmlRequest">
    <xs:complexContent mixed="false">
      <xs:extension base="ReleaseLockRequest" />
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ReleaseLockRequest">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Username" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="LockString" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.8 Inbetriebnahme der Demo-Applikation

Dieser Abschnitt gibt eine kurze Anleitung zur Inbetriebnahme der Demo-Applikation. Alle erwähnten Dateipfade gehen vom Wurzelverzeichnis des Demo-Projekts aus (dies ist jenes Verzeichnis, in welchem sich die Datei *OFDP-Demo_v2.sln* befindet). Diese Dateipfade beginnen daher immer mit *<Demo-Applikation>*\....
Zur Inbetriebnahme der Demo-Applikation wird folgende Software benötigt:

- Microsoft SQL Server 2008 Express (inkl. Management Tools und Microsoft SQL Server 2008 Management Studio Express)[3]
- Microsoft .NET Framework 3.5[4]
- Microsoft Chart Controls for Microsoft .NET Framework 3.5[5]
- Language Pack für Microsoft Chart Controls für Microsoft .NET Framework 3.5[6]

Weiters müssen die Ports *8010* bis *8016* frei sein. Über diese Ports sind die Services der Serverkomponenten zu erreichen. Sollten diese Ports nicht frei sein, können diese einfach über die Konfigurationsdateien der Serverhosts umkonfiguriert werden.

Vorgangsweise bei der Inbetriebnahme:

1. Als erstes müssen das *Microsoft .NET Framework 3.5* (inklusive der beiden Zusatzpakete) und *Microsoft SQL Server 2008 Express Edition* heruntergeladen und installiert werden. Als Authentifizierungsmodus für die Datenbank sollte der *gemischte Modus* gewählt werden.
2. Als nächstes wird die Clientapplikation des SQL Servers namens *SQL Server 2008 Management Studio Express* gestartet.
3. Danach werden drei neue Datenbanken angelegt. Eine für die Schlüsselverwaltung und je eine für die beiden Datenspeicher. In der ursprünglichen Implementierung wurden diese folgendermaßen benannt:
   - OFDP_KeyManagement_v2
   - OFDP_DataStorage1_v2
   - OFDP_DataStorage2_v2

   Natürlich können die Datenbanken individuell benannt werden. Die entsprechenden Namen müssen anschließend in den Konfigurationsdateien der Services angepasst werden.

[3] erhältlich unter `http://www.microsoft.com/express/Database/`
[4] erhältlich unter `http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=de`
[5] erhältlich unter `http://www.microsoft.com/downloads/details.aspx?FamilyId=130F7986-BF49-4FE5-9CA8-910AE6EA442C&displaylang=en`
[6] erhältlich unter `http://www.microsoft.com/downloads/details.aspx?displaylang=de&FamilyID=581ff4e3-749f-4454-a5e3-de4c463143bd`

4. Anschließend werden die drei in Abschnitt A.9 gezeigten Skripts ausgeführt. Diese Skripts erzeugen alle notwendigen Tabellen in den soeben angelegten Datenbanken. Falls andere Namen für die Datenbanken gewählt wurden, müssen diese in den Skripts angepasst werden.

5. Als nächstes müssen in folgenden Dateien die Connection-Strings mit der Verbindung zur entsprechenden Datenbank angepasst werden:

   - *<Demo-Applikation>\KeyManagementHost\bin\Release\KeyManagementHost.exe.config*
   - *<Demo-Applikation>\DataStorageHost1\bin\Release\DataStorageHost1.exe.config*
   - *<Demo-Applikation>\DataStorageHost2\bin\Release\DataStorageHost2.exe.config*

6. Danach können die Server-Komponenten gestartet werden. Dazu werden folgende Dateien mit Administratorrechten ausgeführt:

   - *<Demo-Applikation>\KeyManagementHost\bin\Release\KeyManagementHost.exe*
   - *<Demo-Applikation>\DataStorageHost1\bin\Release\DataStorageHost1.exe*
   - *<Demo-Applikation>\DataStorageHost2\bin\Release\DataStorageHost2.exe*

7. Anschließend können die Clients gestartet werden. Dazu werden folgende Dateien mit Administratorrechten ausgeführt:

   - *<Demo-Applikation>\AutomaticMeter\bin\Release\AutomaticMeter.exe*
   - *<Demo-Applikation>\ClientBerater\bin\Release\ClientBerater.exe*
   - *<Demo-Applikation>\ClientKunde\bin\Release\ClientKunde.exe*

Nun wurden alle notwendigen Komponenten eingerichtet und gestartet.

A.9 Listings der Skripts zum Erstellen der Datenbanktabellen

Die folgenden Listings zeigen Skripts zur Erstellung der Datenbanktabellen der implementierten Demo-Applikation für eine Microsoft SQL Express Datenbank.

A.9.1 Tabellen für die *Schlüsselverwaltung*

```
1  USE [OFDP_KeyManagement_v2]
2  GO
3  SET ANSI_NULLS ON
4  GO
5  SET QUOTED_IDENTIFIER ON
6  GO
7  SET ANSI_PADDING ON
8  GO
9  CREATE TABLE [dbo].[user](
10         [userID] [int] IDENTITY(1,1) NOT NULL,
11         [userName] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
12         [lastWriteTime] [datetime] NULL,
13         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
14         [lastWriteUserID] [int] NULL,
15  CONSTRAINT [PK_user] PRIMARY KEY CLUSTERED
16 (
17         [userID] ASC
18 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
19  CONSTRAINT [unique_username] UNIQUE NONCLUSTERED
20 (
21         [userName] ASC
22 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
23 ) ON [PRIMARY]
24
25
26 CREATE TABLE [dbo].[dataStorage](
27         [dataStorageID] [int] IDENTITY(1,1) NOT NULL,
28         [linkAddress] [varchar](255) COLLATE Latin1_General_CI_AS NOT NULL,
29         [name] [varchar](255) COLLATE Latin1_General_CI_AS NOT NULL,
30         [lastWriteTime] [datetime] NULL,
31         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
32         [lastWriteUserID] [int] NULL,
33  CONSTRAINT [PK_dataStorage] PRIMARY KEY CLUSTERED
34 (
35         [dataStorageID] ASC
36 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
37  CONSTRAINT [UQ__dataStorage__014935CB] UNIQUE NONCLUSTERED
38 (
39         [linkAddress] ASC
40 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
41 ) ON [PRIMARY]
42
43
44 CREATE TABLE [dbo].[keyband](
45         [keybandID] [int] IDENTITY(1,1) NOT NULL,
46         [userID] [int] NOT NULL,
47         [lastWriteTime] [datetime] NULL,
48         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
49         [lastWriteUserID] [int] NULL,
50         [algorithm] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
51         [dataStorageID] [int] NULL,
52  CONSTRAINT [PK_keyband] PRIMARY KEY CLUSTERED
53 (
54         [keybandID] ASC
55 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
56 ) ON [PRIMARY]
57
58 ALTER TABLE [dbo].[keyband]  WITH CHECK ADD  CONSTRAINT [FK_keyband_dataStorage] FOREIGN KEY([
       dataStorageID])
59 REFERENCES [dbo].[dataStorage] ([dataStorageID])
60 ALTER TABLE [dbo].[keyband]  WITH CHECK ADD  CONSTRAINT [FK_keyband_user] FOREIGN KEY([userID
       ])
61 REFERENCES [dbo].[user] ([userID])
62
63
64 CREATE TABLE [dbo].[key](
```

```
65          [keyID] [int] IDENTITY(1,1) NOT NULL,
66          [userID] [int] NOT NULL,
67          [keybandID] [int] NULL,
68          [publicKey] [varchar](1000) COLLATE Latin1_General_CI_AS NOT NULL,
69          [privateKey] [varchar](1000) COLLATE Latin1_General_CI_AS NOT NULL,
70          [dataStorageID] [int] NULL,
71          [purpose] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
72          [weekInYear] [int] NULL,
73          [year] [int] NULL,
74          [lastWriteTime] [datetime] NULL,
75          [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
76          [lastWriteUserID] [int] NULL,
77          [algorithm] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
78   CONSTRAINT [PK_key] PRIMARY KEY CLUSTERED
79  (
80          [keyID] ASC
81  )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
82  ) ON [PRIMARY]
83
84  ALTER TABLE [dbo].[key]  WITH CHECK ADD  CONSTRAINT [FK_dataStorage] FOREIGN KEY([
        dataStorageID])
85  REFERENCES [dbo].[dataStorage] ([dataStorageID])
86  ALTER TABLE [dbo].[key]  WITH CHECK ADD  CONSTRAINT [FK_key_keyband] FOREIGN KEY([keybandID])
87  REFERENCES [dbo].[keyband] ([keybandID])
88  ALTER TABLE [dbo].[key]  WITH CHECK ADD  CONSTRAINT [FK_user] FOREIGN KEY([userID])
89  REFERENCES [dbo].[user] ([userID])
90
91
92
93  CREATE TABLE [dbo].[sharingRequest](
94          [userId] [int] NOT NULL,
95          [keyId] [int] NOT NULL,
96          [lastWriteTime] [datetime] NULL,
97          [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
98          [lastWriteUserID] [int] NULL,
99   CONSTRAINT [PK_sharingRequest] PRIMARY KEY CLUSTERED
100 (
101         [userId] ASC,
102         [keyId] ASC
103 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
104 ) ON [PRIMARY]
105
106 ALTER TABLE [dbo].[sharingRequest]  WITH CHECK ADD  CONSTRAINT [FK_sharingRequest_key] FOREIGN
        KEY([keyId])
107 REFERENCES [dbo].[key] ([keyID])
108 ALTER TABLE [dbo].[sharingRequest]  WITH CHECK ADD  CONSTRAINT [FK_sharingRequest_user]
        FOREIGN KEY([userId])
109 REFERENCES [dbo].[user] ([userID])
110
111
112 CREATE TABLE [dbo].[accessControlList](
113         [userID] [int] NOT NULL,
114         [keyID] [int] NOT NULL,
115         [lastWriteTime] [datetime] NULL,
116         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
117         [lastWriteUserID] [int] NULL,
118  CONSTRAINT [PK_accessControlList] PRIMARY KEY CLUSTERED
119 (
120         [userID] ASC,
121         [keyID] ASC
122 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
123 ) ON [PRIMARY]
124
125 GO
126 SET ANSI_PADDING OFF
127 GO
128 ALTER TABLE [dbo].[accessControlList]  WITH CHECK ADD  CONSTRAINT [FK_key] FOREIGN KEY([keyID
        ])
129 REFERENCES [dbo].[key] ([keyID])
130 ALTER TABLE [dbo].[accessControlList]  WITH CHECK ADD  CONSTRAINT [FK_user_ACL] FOREIGN KEY([
        userID])
131 REFERENCES [dbo].[user] ([userID])
132
133
134 insert into dbo.[user] (userName) values ('all');
135 insert into dbo.dataStorage (linkAddress, [name]) values ('http://localhost:8011/DataStorage',
        'DataStorage1');
```

```
136 insert into dbo.dataStorage (linkAddress, [name]) values ('http://localhost:8012/DataStorage',
        'DataStorage2');
137
138 GO
```

A.9.2 Tabellen für *Datenspeicher 1*

```
1  USE [OFDP_DataStorage1_v2]
2  GO
3
4  SET ANSI_NULLS ON
5  GO
6  SET QUOTED_IDENTIFIER ON
7  GO
8  SET ANSI_PADDING ON
9  GO
10 CREATE TABLE [dbo].[user](
11         [userID] [int] IDENTITY(1,1) NOT NULL,
12         [userName] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
13         [lastWriteTime] [datetime] NULL,
14         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
15         [lastWriteUserID] [int] NULL,
16  CONSTRAINT [PK_user] PRIMARY KEY CLUSTERED
17 (
18         [userID] ASC
19 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
20  CONSTRAINT [unique_username] UNIQUE NONCLUSTERED
21 (
22         [userName] ASC
23 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
24 ) ON [PRIMARY]
25
26 CREATE TABLE [dbo].[storage](
27         [ID] [int] IDENTITY(1,1) NOT NULL,
28         [userID] [int] NOT NULL,
29         [type] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
30         [encryptedData] [image],
31         [encryptionAlgorithm] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
32         [lastWriteTime] [datetime] NULL,
33         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
34         [lastWriteUserID] [int] NULL,
35  CONSTRAINT [PK_storage] PRIMARY KEY CLUSTERED
36 (
37         [ID] ASC
38 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
39 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
40
41 CREATE TABLE [dbo].[keysForStorage](
42         [ID] [int] NOT NULL,
43         [KeyID] [int] NOT NULL,
44         [encryptedSessionKey] [image] NOT NULL,
45         [encryptionAlgorithm] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
46  CONSTRAINT [PK_keysForStorage] PRIMARY KEY CLUSTERED
47 (
48         [ID] ASC,
49         [KeyID] ASC
50 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
51 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
52
53 alter table dbo.storage add constraint FK_user foreign key (userID)
54 references dbo.[user] (userID);
55
56 alter table dbo.keysForStorage add constraint FK_ID foreign key (ID)
57 references dbo.storage (ID);
58
59 GO
60 SET ANSI_PADDING OFF
```

A.9.3 Tabellen für *Datenspeicher 2*

```
1  USE [OFDP_DataStorage2_v2]
2  GO
3
4  SET ANSI_NULLS ON
5  GO
```

```
6  SET QUOTED_IDENTIFIER ON
7  GO
8  SET ANSI_PADDING ON
9  GO
10 CREATE TABLE [dbo].[user](
11         [userID] [int] IDENTITY(1,1) NOT NULL,
12         [userName] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
13         [lastWriteTime] [datetime] NULL,
14         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
15         [lastWriteUserID] [int] NULL,
16  CONSTRAINT [PK_user] PRIMARY KEY CLUSTERED
17 (
18         [userID] ASC
19 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
20  CONSTRAINT [unique_username] UNIQUE NONCLUSTERED
21 (
22         [userName] ASC
23 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
24 ) ON [PRIMARY]
25
26 CREATE TABLE [dbo].[storage](
27         [ID] [int] IDENTITY(1,1) NOT NULL,
28         [userID] [int] NOT NULL,
29         [type] [varchar](50) COLLATE Latin1_General_CI_AS NOT NULL,
30         [encryptedData] [image],
31         [encryptionAlgorithm] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
32         [lastWriteTime] [datetime] NULL,
33         [lastOperation] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
34         [lastWriteUserID] [int] NULL,
35  CONSTRAINT [PK_storage] PRIMARY KEY CLUSTERED
36 (
37         [ID] ASC
38 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
39 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
40
41 CREATE TABLE [dbo].[keysForStorage](
42         [ID] [int] NOT NULL,
43         [KeyID] [int] NOT NULL,
44         [encryptedSessionKey] [image] NOT NULL,
45         [encryptionAlgorithm] [varchar](50) COLLATE Latin1_General_CI_AS NULL,
46  CONSTRAINT [PK_keysForStorage] PRIMARY KEY CLUSTERED
47 (
48         [ID] ASC,
49         [KeyID] ASC
50 )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
51 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
52
53
54
55 alter table dbo.storage add constraint FK_user foreign key (userID)
56 references dbo.[user] (userID);
57
58 alter table dbo.keysForStorage add constraint FK_ID foreign key (ID)
59 references dbo.storage (ID);
60
61 GO
62 SET ANSI_PADDING OFF
```

Literaturverzeichnis

[1] 3. EU-Binnenmarktspaket. http://www.e-control.at/portal/pls/portal/portal.kb_folderitems_xml.redirectToItem?pMasterthingId=273223 - Stand 6.3.2010.

[2] Comments of Google on Smart Grid Technology Development in California. http://www.google.org/powermeter/cpuc.html - Stand: 20.11.2009.

[3] Erklärung des Begriffs: Windows Communication Foundation (WCF) - Was ist Windows Communication Foundation (WCF)? http://www.it-visions.de/glossar/alle/3719/lexikon.aspx - Stand: 8.2.2010.

[4] Verschlüsselung (symmetrisch, asymmetrisch, hybrid). http://www.tcp-ip-info.de/security/verschluesselung.htm - Stand: 11.2.2010.

[5] Michele Leroux Bustamante. *Learning WCF*. O'Reilly, Mai 2007.

[6] Microsoft Corporation. Service Versioning. http://msdn.microsoft.com/en-us/library/ms731060.aspx - Stand: 8.2.2010.

[7] Microsoft Corporation. Windows Communication Foundation. http://msdn.microsoft.com/en-us/netframework/aa663324.aspx - Stand: 8.2.2010.

[8] Achim Demelt. Temporale Datenhaltung. *JavaSPEKTRUM*. Ausgabe 5/2003 - http://www.exxcellent.de/download/demelt_JS_05_03.pdf - Stand: 6.2.2010.

[9] Energie-Control GmbH. Smart Metering. http://www.e-control.at/de/marktteilnehmer/strom/smart-metering - Stand: 6.3.2010.

[10] Franz Schodl (PDTS GmbH). White Book - Open Facility Data Platform. http://www.pdts.at/cms/upload/download/WhiteBook-OpenFacilityDataPlatform-12.pdf - Stand: 15.2.2010.

[11] heise online. Geplantes neues US-Stromnetz anfälliger für Angriffe. http://www.heise.de/security/meldung/Geplantes-neues-US-Stromnetz-anfaelliger-fuer-Angriffe-210265.html.

[12] Torsten Horn. Web Services mit SOAP (Simple Object Access Protocol). http://www.torsten-horn.de/techdocs/soap.htm#WebServices - Stand 9.3.2010.

[13] TU Berlin Informatik und Gesellschaft. Einweg-Hashfunktion. http://www.ig.cs.tu-berlin.de/oldstatic/ap/rg/002/glossar/e-terms/einweghashfunktion.html - Stand: 12.3.2010.

[14] IOActive. IOActive verifies critical flaws in next generation energy infrastructure. http://www.ioactive.com/news-events/AMIAlertPR.html - Stand: 12.3.2010.

[15] Candace Lombardi. Google Powermeter available to U.K. residents. http://news.cnet.com/8301-11128_3-10385219-54.html - Stand: 19.11.2009.

[16] Clara Marshall. Google PowerMeter. http://www.davidnaylor.co.uk/google-powermeter.html - Stand: 19.11.2009.

[17] Sascha Möllering. Historisierung mit Oracle und Hibernate. http://it-republik.de/jaxenter/artikel/Historisierung-mit-Oracle-und-Hibernate-2621.html - Stand: 6.2.2010.

[18] Daniel Ohst. Einsatz elektronischer Signaturen und Zeitstempel für die Sicherung digitaler Dokumente. Master's thesis, Humboldt-Universität zu Berlin, 2003. http://edoc.hu-berlin.de/master/ohst-daniel-2004-04-20/HTML/front.html, Kapitel 3, Datum: 1.9.2003.

[19] Josef Stocker. Smart Metering-der intelligente Stromzähler. http://oekocenter.bewag.at/energiesparen/stories/35572/ - Stand: 6.3.2010.

[20] Jim Stogdill. Google's Powermeter. It's Cool, but don't Bogart My Meter Data. http://radar.oreilly.com/2009/02/googles-powermeter-its-cool-bu.html - Stand: 19.11.2009.

[21] David Booth und Andere. Web Service Architecture. http://www.w3.org/TR/ws-arch/#whatis - Stand: 8.3.2010.

[22] Martin Schultz und Claudia Hübner und Julia Zinke. Smart Metering-Innovation in Technologie und Abrechnung!? http://www.e-journal-of-pbr.info/wiki/index.php/Smart_Metering - Stand: 6.3.2010.

[23] Kyle Brown und Michael Ellis. Best practices for web services versioning. http://www.ibm.com/developerworks/webservices/library/ws-version/ - Datum: 30.1.2004.

[24] W3C. SOAP Version 1.2. http://www.w3.org/TR/soap12/ - Stand: 9.3.2010.

[25] W3Schools. SOAP Tutorial. `http://www.w3schools.com/soap/default.asp` - Stand: 9.3.2010.

[26] Yello Strom GmbH. `http://google.yellostrom.de` - Stand: 20.11.2009.