DB Dienste zur Unterstützung des LocationMaster als Trustpoint

DBAP9

15.12.2014, Henning Budde

Andreas Lockermann

Thomas Partsch

1. DB Dienste zur Unterstützung des LocationMaster als TrustPoint

Dieses Dokument ist der Bericht zum Arbeitspaket DBAP9. Mit dem Arbeitspaket werden folgende zwei Ziele angestrebt:

1. Entwicklung einer auf User und Gruppen bezogenen Verwaltung von Sensordaten, die in einem Sensornetz, das über einem LocationMaster als Trust Point mit der Cloud verbunden ist, erhoben werden. Insbesondere soll diese auf User und Gruppen bezogene Verwaltung der Sensordaten im FDBS Schema abgebildet werden.
2. Entwicklung von DBS Programmen auf dem LocationMaster und in der Cloud zur datenbanktechnischen Umsetzung der von Konsortium auf Basis eines Benutzerrollenmodells entwickelten Trust Point Konzepten.

Der Stand der Ergebnisse zu beiden Zielen wird nachfolgend dargestellt.

2. Auf Nutzer und Gruppen bezogene Verwaltung von Sensordaten

Im Folgenden wird der auf User und Gruppen bezogene Ausschnitt des FDBS Schemas dargestellt. Aufgrund der in Meilenstein 1 (vgl. [1]) analysierten Benutzerrollen potentieller User der SensorCloud wurde ein allgemeines Nutzer-Konzept für das FDBS-Schema entwickelt. Ein User kann als Person oder technische Entität Nutzer der SensorCloud sein. Jeder Nutzer ist als Instanz der Entität „NutzerStammdaten“ im FDBS gespeichert. Ein Nutzer kann zusammen mit Nutzern, die eine ähnliche Benutzerrolle haben, in einer Gruppe verwaltet werden. Um die auf Nutzer und Gruppen bezogene Verwaltung von Sensordaten strukturiert im FDBS darstellen zu können, wurde dafür ein besonderes Entity-Relationship-Modell entwickelt, das in nachfolgender Abbildung 1 gegeben ist. In der Abbildung ist zu sehen, dass ein Nutzer (Entität NutzerStammdaten) „Besitzer“ von Aktoren, Sensoren, Lokationen und Gateways (LocationMaster) ist. Demnach gehört jeder Aktor, jeder Sensor, jeder LokationMaster zu einem Nutzer. Diese Zuordnung ist auch in Anlehnung an das Benutzermodell ([1]) zu sehen: Dabei gibt es z.B. eine Benutzerrolle „TP Admin“ (Trust Point), die das Recht hat einen Trust Point zu verwalten. Ein Trustpoint wird typischerweise auf einem Rechner vom Typ LocationMaster eingerichtet. Eine weitere Benutzerrolle ist z.B. die Rolle „SensorOwner“ die Besitzer von Sensoren und Aktoren ist.

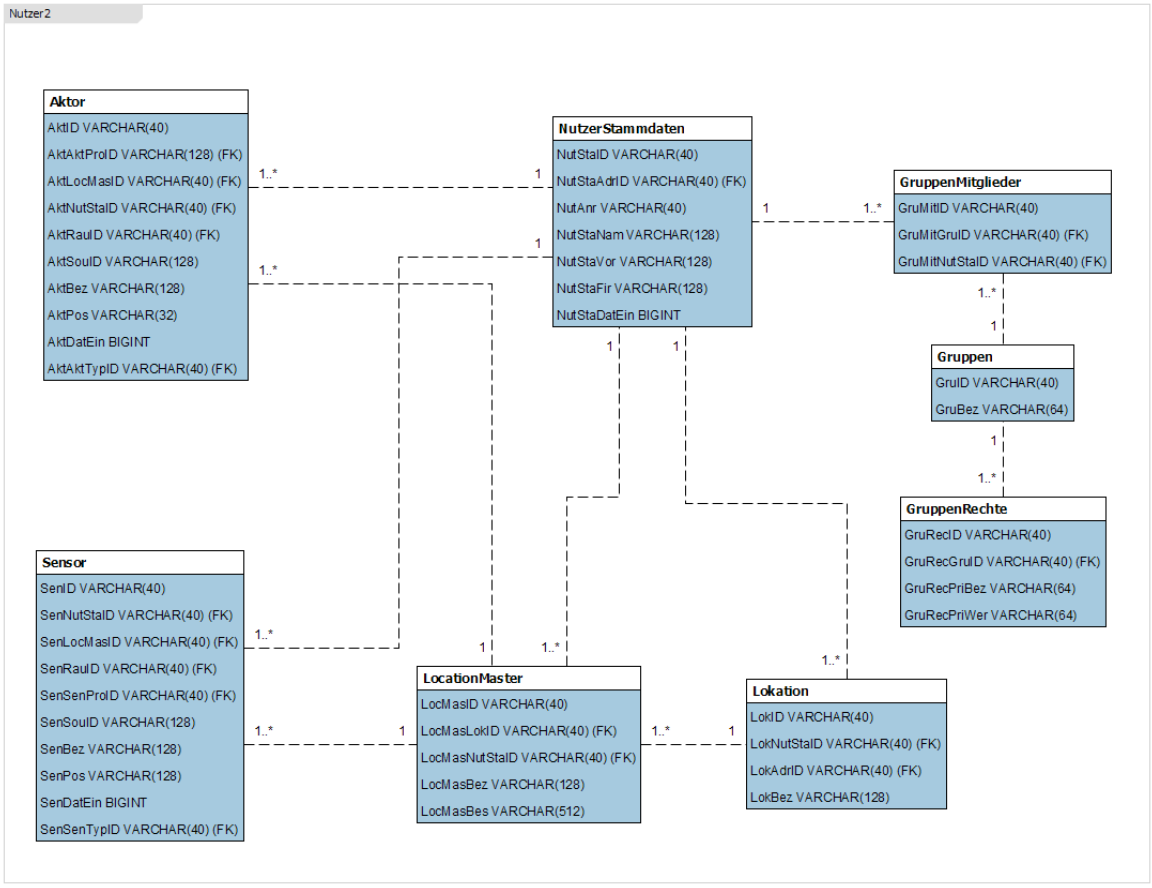

Abbildung 1: Ausschnitt I aus dem konzeptionellen Schema, Quelle: konzeptionelles Schema (FDBS FH Köln)

Ein Nutzer (Entität NutzerStammdaten) kann einer (oder mehreren) Gruppen angehören, eine Gruppe besitzt verschiedene Rechte (Entität GruppenRechte). Zum Beispiel kann es eine Gruppe geben, in der nur Nutzer enthalten sind, die einer Benutzerrolle angehören.

Ein Nutzer hat die Möglichkeit, mehrere EMail-Adressen und Telefonnummern in der Datenbank abzulegen. Für weitere Informationen, die dem Nutzer zugeordnet sind, gibt es weitere Entitäten, diese sind in Abbildung 3 dargestellt. Es gibt z.B. die Entitäten NutzerEmail und NutzerTelefon. Dort können weitere einem Nutzer zugeordnete Informationen abgelegt werden. In der Entität NutzerSicherheit können z.B. Schlüssel eines Nutzers abgelegt werden.

Entität Nutzersicherheit:

Name	Datentyp	Beschreibung	Beziehung / Integrität
NutSicID	text	Nutzersicherheit ID Eindeutige ID für die Entität Sicherheit eines Nutzers	PRIK
NutSicNutStaID	text	Nutzerstammdaten ID ID eines Nutzers (-stammdaten)	FKEY: NutzerStammdaten.NutStaID
NutSicPas	text	Nutzersicherheit Passwort Passwort eines Nutzers (Hash des Passwortes (z.B. mit SHA-2, scrypt o.ä.)	/
NutSicPriKey	text	Nutzersicherheit Private Key	/

		Private Key eines Nutzers	
NutSicPubKey	text	Nutzersicherheit Public Key Public Key eines Nutzers	/

Abbildung 2: Entität NutzerSicherheit, Quelle: konzeptionelles Schema (FDBS FH Köln)

Für jeden Nutzer (referenziert zu der Entität NutzerStammdaten) können ein Passwort für den Login sowie weitere Schlüssel (z.B. Private Key) abgelegt werden. Die folgende Abbildung zeigt den Nutzer im Zusammenhang mit anderen Entitäten im konzeptionellen Schema. Über das Attribut „NutStaAdrID“ kann dem Nutzer eine Adresse zugeordnet werden (hier nicht abgebildet). Mit der Ausgliederung von dem Nutzer zugeordneten Daten wie z.B. E-Mail oder Telefon können einem Nutzer mehrere Informationen des gleichen Typs zugeordnet werden. So kann ein Nutzer z.B. mehrere Telefonnummern besitzen.

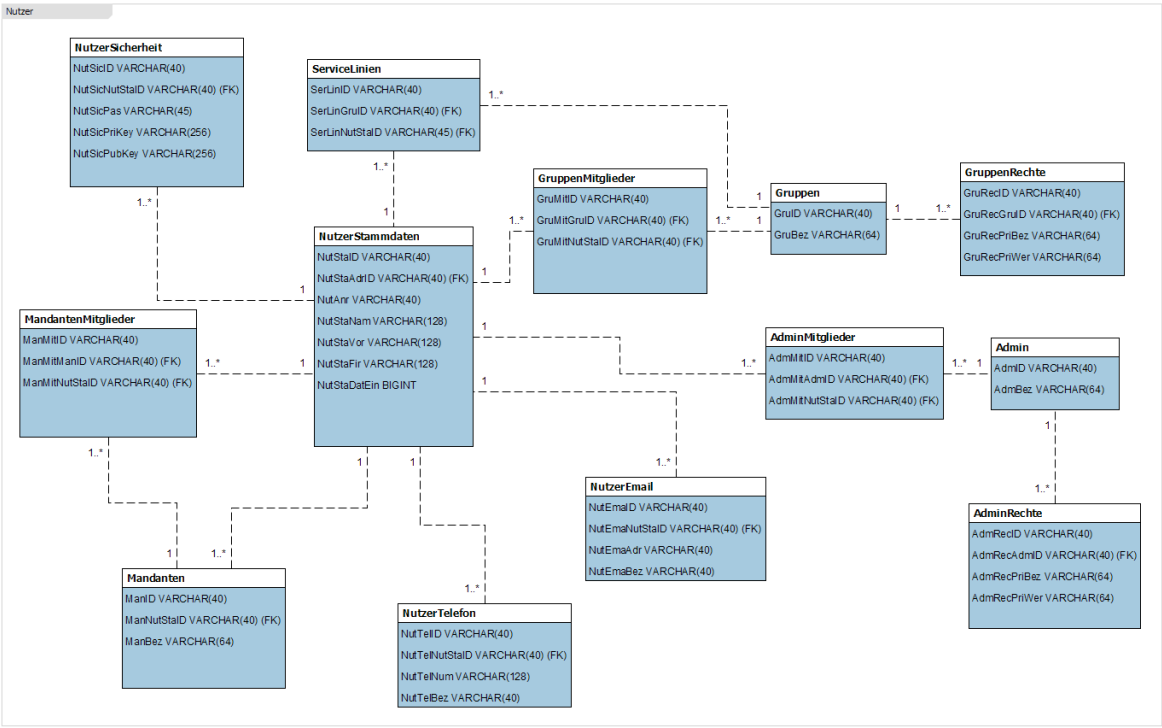

Abbildung 3: Ausschnitt II aus dem konzeptionellen Schema,, Quelle: konzeptionelles Schema (FDBS FH Köln)

Die wesentlichen Bestandteile von SensorCloud sind die logischen Komponenten „Iaas Cloud“, „SensorCloud Plattform / Cloud Service Plattform (PaaS)“, „Sensor Netzwerk / Sensor-Gateway“ und das „Frontend“ ([1]). Das Benutzermodell (vgl. [1], Fokussierung auf den UseCase Weitgehende Home-Automation einer Familie der „bürgerlichen Mitte“) sieht folgende „Technische Entitäten“ / „Benutzerrollen“ vor [1]:

- IaaS Cloud / IaaS Provider: IaaS Provider stellt IaaS Cloud zur Verfügung.
- PaaS Cloud / PaaS Provider: Paas Cloud wird vom Paas Provider zur Verfügung gestellt.
- Service / Service Provider: Die Services, die auf der Plattform zur Verfügung gestellt werden vom Service Provider zur Verfügung gestellt.
- Service Instanz / Service Owner: Ein in der Cloud ausgeführter Service wird als Service Instanz bezeichnet. Der Besitzer der Service Instanz wird Service Owner bezeichnet.
- Service User: Ein Software-Agent oder eine Person die sich anmelden muss. Ein Service User kann einen Service nutzen. Der Service Owner legt hierbei Berechtigungen des Service Users fest.
- Sensor / Aktor: Bestimmte Eigenschaften in Umgebungen werden vom Sensor gemessen bzw. erfasst. Elektrische Signale werden von einem Aktor in mechanische Bewegungen oder

physikalische Wirkungen umgesetzt. Zusätzlich gibt es noch virtuelle Sensoren und Aktoren.

- Sensor Netzwerk / Sensor Owner: Einem Sensor Owner sind Sensoren und Aktoren zugeordnet, die sich in einem Sensor Netzwerk befinden. Dem Sensor Owner gehören die Daten, die vom Sensor erzeugt worden sind und nur der Sensor Owner darf auf Aktoren zugreifen.
- Sensor Gateway / Trust Point (TP) / TP Admin: Die Kommunikation von Sensoren eines Sensor Netzwerks mit der Cloud erfolgt über das Gateway. Das Gateway fungiert dabei als Trust Point. Dabei werden verschiedene Protokolle der Sensoren und Aktoren vereinheitlicht, die Kommunikation erfolgt dann mit einer Service Instanz. Geeignete Mechanismen für eine sichere Übertragung werden von der Cloud Service Plattform zur Verfügung gestellt. Ein Service User wird für die Kommunikation verwendet und TP Admin verwaltet den Trust Point. Aufgaben des TP Admins sind dabei die Zuordnung von Sensor Owner zu Sensoren und Aktoren.

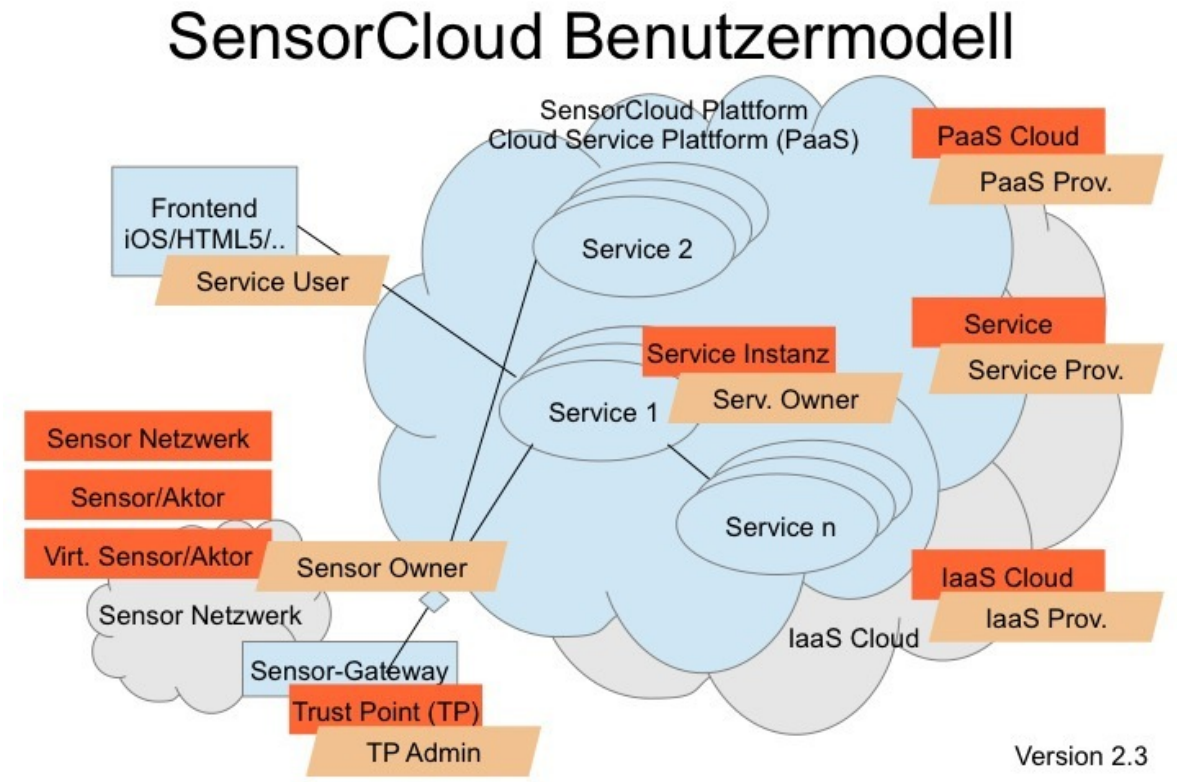

Abbildung 4: Benutzermodell, Quelle: [1]

Sämtliche dieser durch Benutzerrollen charakterisierten Nutzer sind mit ihren Attributen im konzeptionellen Schema der FDBS der SensorCloud abgebildet worden.

3. Trust Point Konzept

Im Projekt SensorCloud spielen die Sicherheit und der Datenschutz eine zentrale Rolle. Aus diesem Grund entwickelte die RWTH Aachen eine Trust-Point-Sicherheitsarchitektur. Ein Trust Point stellt die Anbindung an die Cloud. Die Daten, die lokal von einem Gateway (LocationMaster) an die Cloud weitergegeben werden, werden verschlüsselt. Mit dem Einsatz des Trust Points wird Vertrauen in die SensorCloud geschaffen und die Akzeptanz des Systems gesteigert (vgl. [2]). Jeder Rechner vom Typ LocationMaster besitzt eine Instanz des Trust Points. Die TrustPoint-Architektur (laut Meilenstein 1, [1]) ist in der folgenden Abbildung dargestellt:

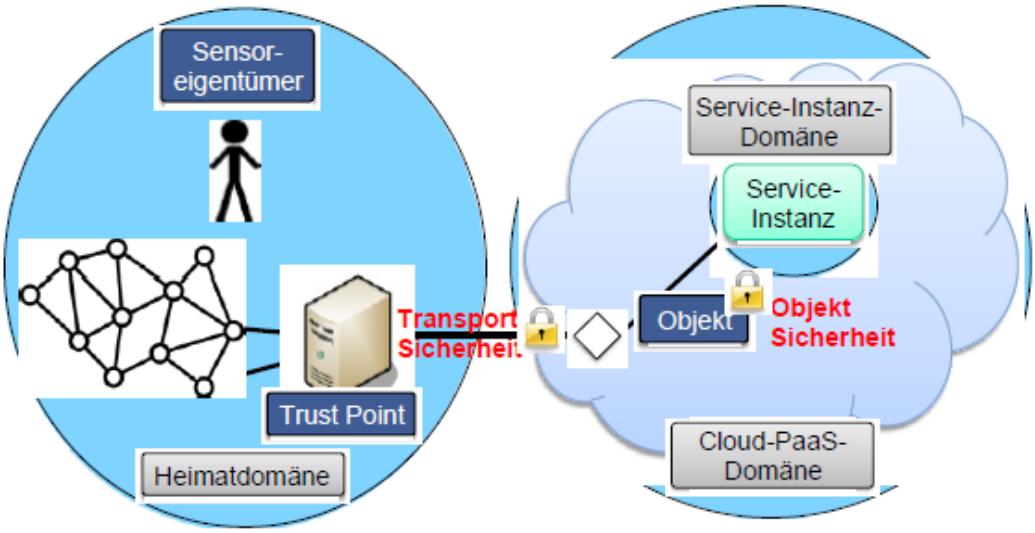

Abbildung 5: Visualisierung der TrustPont-Architektur, Quelle: [1]

Für die SensorCloud werden verschiedene Vertrauensbereiche definiert. In der Abbildung sind davon die Vertrauensbereiche Heimatdomäne, Service-Instanz-Domäne und Cloud-PaaS-Domäne zu sehen. In der Heimatdomäne gibt es das Sensornetzwerk mit dem Gateway. Der Eigentümer der Sensoren darf den Geräten in der Heimatdomäne vertrauen. Sobald die Daten außerhalb des Sensornetzwerks an Entitäten weitergegeben werden, endet der Vertrauensbereich. Wenn Daten in die Cloud vom Sensoreigentümer ausgelagert werden, ergeben sich drei zentrale Anforderungen [1]:

- Nur in den Bereichen, denen der Sensoreigentümer vertraut, darf die Verarbeitung und Speicherung der Sensor-/Aktor-Daten durchgeführt werden.
- Ohne eine explizite Freigabe können Dritte nicht auf Sensor-/Aktor-Daten zugreifen oder diese manipulieren.
- Welche Service-Instanzen Zugriff auf Sensor-/Aktor-Daten erhalten, entscheidet der Sensoreigentümer.

Als Kontrollinstanz an der Netzwerkgrenze dient der Trust Point, er agiert sozusagen als „Stellvertreter“ des Sensoreigentümers. Das Sicherheitskonzept für SensorCloud ist in [5], die „Auswahl von Sicherheitsmechanismen (Objekt- und Transportsicherung im Rahmen der Sicherheitsarchitektur“ ist in [6] und die „Objektsicherheit und Zugriffskontrollmechanismen“ sind in [7] beschrieben.

Der Trust Point ist zurzeit unter anderem auf dem LocationMaster an der FH Köln (AG Testbett FH Köln) implementiert. Der Aufbau der Architektur des LocationMasters im Testbett der FH Köln ist in folgender Abbildung (Abbildung 6, Fokussierung im Folgenden auf den Trust Point) dargestellt:

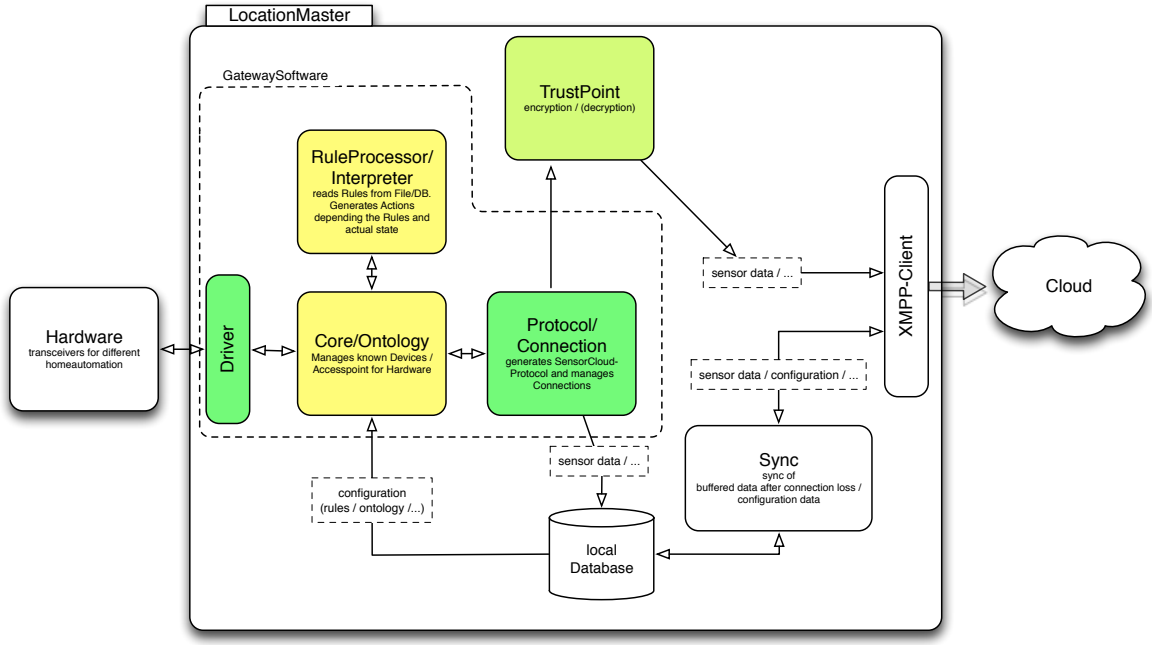

Abbildung 6: Architektur LocationMaster Testbett FH-Köln, Quelle: Testbett FH Köln

Die Hardware liefert Messwerte in Gestalt von Rohdaten verschiedener Sensoren, die dann vom jeweiligen Treiber auf das SensorProdukt bezogen interpretiert (Semantische Analyse mittels Ontologien, vgl. [3]) werden und dann im JSON-Format mittels des Trust Points verschlüsselt werden. Der Trust Point läuft als eigenständige Software, die über Unix-Pipes kommuniziert bzw. gesteuert wird. In eine Pipe "data.input" werden zeilenweise Payload-Objekte nach der Spezifikation für SensorData (JSON-Darstellung, vgl. [4]) geschrieben. Diese Objekte verschlüsselt der Trustpoint je nach Konfiguration des einzelnen Sensors und des jeweils zugeordneten SensorCloud-Services. Das verschlüsselte JSON-Objekt schreibt der TrustPoint danach wieder in eine Pipe "data.output“. Über eine dritte Pipe "DataProcessor.command_fifo" kann der Trust Point gesteuert bzw. konfiguriert werden. Derzeit ist hier zu Demonstrationszwecken ein per Webbrowser manuell steuerbares Webinterface „TrustPoint Konfiguration“ angekoppelt, über welches Demo-Konfigurationen verändert werden können. Für einen produktiven Einsatz des Trustpoints bzw. LocationMasters muss die in der SensorCloud hinterlegte Konfiguration eingelesen und dem TrustPoint über die „DataProcessor.command_fifo“-Pipe bekannt gemacht werden.

```
{
   "typ":"1",
   "gw":"string",
   "bn":"string",
   "bt":"number",
   "e":[
   {
      "n":"Temperatur",
      "sv":"ab1"
},
   {
      "n":"Luftfeuchte",
      "sv":"ce8"
   },
   {
      "n":"Bewegung",
      "sv":"1"
   }
   ]
}
```

```
{
   "typ":"1",
   "gw":"string",
   "bn":"string",
   "bt":"number",
   "e":[
   {
      "n":"Temperatur",
      "ev":[
      {
         "unprotected":
         {
            "alg":"dir",
            "enc":"AESGCM256",
             "kid":"851acaf912c08
                    3843549508e7d411fb73a7a8c5b",
            "typ":"sv"
         },
         "iv":" AQABAAEAAAAAAAAAAAAAAA==",
         "ciphertext":" AvFvx6kAPd1V",
         "tag":" NpOsZHe8Ys+RFSQ8vbTD5g=="
      }
      ]
   },
   {
      "n":"Luftfeuchte",
      "sv":"ce8"
   },
   {
      "n":"Bewegung",
      "sv":"1"
   }
   ],
 "sig":
 {
   "signatures":[
   {
      "header":
      {
         "alg":"ES256"
      },
         "signature":" MC0CFG2YmIu64ltOuPH2otrO1e7

/xOHsAhUAyAQbkP6JdzIfE/wsF8WKCn936eQ="
   }
   ]
 }
}
```

Abbildung 7: JSON-Payload (Beispiel), Quelle: Testbett FH Köln

In der Abbildung 7 ist die Darstellung eines Messwerts vor (linke Seite) und nach (rechte Seite) der Verschlüsselung durch den Trust Point dargestellt.

Der Trustpoint generiert in definierbaren periodischen Zeitabständen symmetrische kryptographische Schlüssel für einen verschlüsselten Messwertversand. Diese Schlüssel werden mit den kryptographischen Public Keys - eines asymmetrischen kryptographischen Verfahrens - der die Messwerte nutzenden Services verschlüsselt. Die verschlüsselten symmetrischen Schlüssel werden in SensorCloud-Nachrichten vom Typ 400 [8] in die SensorCloud übertragen.

Die für die Verschlüsselung benötigten Public Keys der Services werden aus Richtung Cloud zu den LocationMastern synchronisiert und bei Systemstart des lokalen Trustpoints sowie in periodischen Zeitabständen in den Speicher des Trustpoints geladen.

Das folgende Listing beschreibt eine ausgehende Typ-400-Nachricht.

```
{
```

```
    "typ": "400",
    "service_id": "STRING",  /* targeted unique service ID */
    "gw": "STRING",  /* unique gateway ID */
    "bn": "STRING",  /* sensor device ID */
    "json_path": "STRING", /* used for data item */
    "timerange_start": "STRING", /* valid for timerange - start */
    "timerange_end": "STRING", /* valid for timerange - end */
    "key": {
        "kty": "oct", /* fixed string */
        "kid": "STRING", /* symmetric key ID */
        "alg": "STRING", /* cryptographic algorithm */
        "ev": [
            {
                "unprotected": {
                    "alg":  "RSA1_5",  /*  fixed  string  (cryptographic
                                          primitive and padding scheme) */
                    "enc": "dir", /* fixed string */
                    "kid": "STRING", /* public key ID oft he authorized
                                          service */
                    "typ": "k" /* fixed string */
                },
                "ciphertext": "STRING" /* encrypted symmetric key */
            }
        ]
    }
}
```

Listing 1: Typ-400 Nachricht

Der Nachrichtentyp 400 enthält mit der Angabe der service_id die Information, für welchen Service der verschlüsselte Schlüssel bestimmt ist. Die Attribute timerange_start und timerange_end spezifizieren den Gültigkeitszeitraum des Schlüssels. Beim Eintreffen der Typ-400-Nachricht in der SensorCloud wird der empfangene Schlüssel im KeyStore der SensorCloud persistiert. Der KeyStore ist aktuell mit MySQL in einem relationalen DBMS implementiert. Aufgrund der FDBS-Deploymentanalyse [9] ist eine spätere Migration zu PostgreSQL als RDBMS erstrebenswert.

Die Speicherung der verschlüsselten symmetrischen Schlüssel erfolgt über einen PUT-Request mit dem Typ-400-Paylods auf die RESTful-API des KeyStores. Listing 2 beschreibt die URI des KeyStores für symmetrische Schlüssel.

```
http://babeauf.nt.fh-koeln.de/SensorCloud/KeyData
```

Listing 2: URI des KeyStores für symmetrische Schlüssel

Autorisierte Services selektieren ihre für die Messwerteentschlüsselung benötigten Schlüssel über einen GET-Request auf die RESTful API des KeyStores (Listing 4). Für die Entschlüsselung von Messwerten stellt die RWTH-Aachen eine C-Library zur Verfügung. Bei Verwendung der Library müssen die nutzenden Services eine Callbackfunktion der Library für die Entschlüsselung implementieren. Listing 3 zeigt die Signatur der zu implementierenden Callbackfunktion.

```
JWK_SYMMETRIC_KEY sym_callback(const char *key_id);
```

Listing 3: Signatur der zu implementierenden Callback-Funktion für die Selektion symmetrischer Keys

Die Library nutzt die Callbackfunktion während der Entschlüsselung um symmetrische Schlüssel nachzuladen. Schlüssel können anhand der Key ID (welche in dem verschlüsselten Messwert hinterlegt ist) und der Service ID des Services über die RESTful-API des KeyStores (Listing 4) selektiert werden. Zusammen mit dem Private Key des Services, welcher beim Starten eines Service geladen wird, und dem verschlüsselten symmetrischen Schlüssel kann ein Service mit der Library verschlüsselte Messwerte entschlüsseln.

```
<Ressource>     ::=     http://babeauf.nt.fh-koeln.de/SensorCloud/KeyData
    "/KeyID/" <KeyID> "/ServiceID/" <ServiceID>
```

```
<KeyID> ::= <ID des symmetrischen Schlüssels>

<ServiceID> ::= <ID des Service>
```

Listing 4: URI einer KeyData-Ressource in Backus-Naur-Form

Die Rückgabe eines GET-Requests auf eine KeyData-Ressource erfolgt im JSON-Format (Listing 5). Die Beschreibung der Attribute entspricht der Beschreibung aus Listing 1.

```
{
    "kty": "oct",
    "kid": "STRING",
    "alg": "STRING"
    "ev": [
        {
            "unprotected": {
                "alg": "RSA1_5",
                "enc": "dir",
                "kid": "STRING",
                "typ": "k"
            },
            "ciphertext": "STRING"
        }
    ],
}
```

Listing 5: KeyData-Ressource im JSON-Format

SensorData-Pakete (SensorCloud-Nachrichten vom Nachrichtentyp Typ 1) werden mit dem Private Key der LocationMaster signiert. Zur Überprüfung der Authentizität empfangener SensorData Pakete auf Cloudseite werden die Public Keys der LocationMaster benötigt. Hierzu ist ein RESTful Service für die Selektion der Public Keys der LocationMaster implementiert.

Listing 6 beschreibt die URI einer Public Key-Ressource eines LocationMasters.

```
<Ressource>    ::=    http://babeauf.nt.fh-koeln.de/SensorCloud/PubKeyData
     "/LocationMasterID/" < LocationMasterID > "/KeyType/" <KeyType>

< LocationMasterID > ::= <ID des LocationMasters>

< KeyType > ::= <Verwendetes kryptograpisches Verfahren>
```

Listing 6: URI einer PubKeyData-Ressource in Backus-Naur-Form

Für die Selektion der bei einer Authentizitätsüberprüfung benötigten Public Keys muss ein Service die Callbackfunktion der Library implementieren (Listing 7). Die Callbackfunktion nutzt die Library zur Selektion der LocationMaster Public Keys. Diese können über die RESTful-API des KeyStores anhand der LocationMaster ID selektiert werden (Listing 6). Da ein LocationMaster gleichzeitig verschiedene asymmetrische kryptographische Verfahren nutzen kann, ist die Angabe des verwendeten Verfahrens obligatorisch.

```
JWK_PUBLIC_KEY pub_callback(const char *key_id, PublicKeyType key_type);
```

Listing 7: Signatur der zu implementierenden Callback-Funktion für die Selektion der Public Keys der LocationMaster

Die Rückgabe des GET-Requests auf eine Public Key-Ressource erfolgt im JSON-Format. Die Rückgabe unterscheidet sich je nach verwendeten asymmetrischen kryptographischen Verfahren. Listing 8 zeigt die Rückgabe unter Verwendung von RSA. Die Beschreibung der kty- und kid-Attribute entspricht der Beschreibung aus Listing 1.

```
{
    "kty": "oct",
    "kid": "STRING",
    "n": "STRING",
    "e": "STRING",
}
```

Listing 8: Public Key-Ressource im JSON-Format bei RSA-Verfahren

Listing 9 zeigt die Rückgabe unter Verwendung von Elliptische Kurven. Die Beschreibung der kty- und kid-Attribute entspricht der Beschreibung aus Listing 1.

```
{
    "kty": "oct",
    "kid": "STRING",
    "x": "STRING",
    "y": "STRING",
}
```

Listing 9: Public Key-Ressource im JSON-Format bei RSA-Verfahren

Literatur

[1] Meilensteinbericht 1: Management Summary zum Statusbericht des SensorCloud-Konsortiums zur Erreichung des Meilensteins M1.

[2] SensorCloud: Vorhabensbeschreibung RWTH Aachen University.

[3] Henning Budde, Andreas Lockermann, Thomas Partsch, Gregor Büchel: „Semantische Analyse mittels Ontolgien“, Arbeitsbericht zum Arbeitspaket DPAP3, 15.01.2014.

[4] Protokolltypen der SensorCloud: „SensorCloud JSON Representation“ (Draft) der Protokoll-AG des Konsortium SensorCloud, Version 0.2, Stand: September 2013.

[5] René Hummen, Martin Henze, Daniel Catrein, Klaus Wehrle: „A Cloud Design for User-controlled Storage and Processing of Sensor Data“, Stand: Januar, 2014 http://www.comsys.rwth-aachen.de/fileadmin/papers/2012/2012-hummen-cloud.pdf

[6] Meilensteinbericht 2: Statusbericht des SensorCloud Konsortiums zur Erreichung des Meilensteins M2.

[7] Meilensteinbericht 3: Statusbericht des SensorCloud Konsortiums zur Erreichung des Meilensteins M3.

[8] René Hummen: „Typ-400-Payload“, internes Dokument

[9] A. Lockermann et al.: „*Deploymentanalyse. DBAP2*“, Fachhochschule Köln, Gruppe FDBS, Köln, 2013, http://bscw.fh-koeln.de/bscw/bscw.cgi/d3618380/%5bA2%5d%20Deploymentanalyse,%20Ergebnisse%20zu%20%5bDBAP2%5d.pdf