Hauptseminar: Konzept eines V2X Application-Level Gateways

Sebastian Szancer, Hamburg University of Applied Sciences

Zusammenfassung*—Moderne Fahrzeuge sind heute Teilnehmer in diversen Netzwerken, von VANETs (Vehicular ad-hoc Networks) bis hin zum Internet. Sie kommunizieren mit anderen Fahrzeugen, der Infrastruktur, wie z.B. Ampeln und mit Services in der Cloud. Diese V2X Kommunikation ist von zentraler Bedeutung für die Einführung von innovativen Funktionen wie "over the air" ECU Software Updates, optimaler Navigation und Routenplanung oder autonomem koordinierten Fahren. Es ist zwingend notwendig, dass die V2X Kommunikation entsprechend abgesichert wird, da sie sicherheitskritische Funktionen umfasst. Die Absicherung geschieht durch ein V2X Security Gateway im Fahrzeug, welches den Fahrzeug-internen Diensten, die mit der Außenwelt kommunizieren, als Proxy dient und sowohl die kryptografische Sicherheit, als auch die Sicherheit auf dem Internet-, Transport- und Application-Layer gewährleistet. Die zentrale Komponente eines solchen V2X Security Gateways ist das V2X Application-Level Gateway, welches die Proxy-Funktion, die kryptografische Sicherheit und Sicherheit auf dem Application-Layer realisiert. Diese Arbeit stellt das Konzept eines solchen V2X Application-Level Gateways für IP-basierten Traffic vor.

I. Einleitung

Moderne Fahrzeuge sind heute Teilnehmer in diversen Netzwerken, von VANETs bis hin zum Internet, womit sie zum Teil des "Internet of Things" (IoT) werden. Sie kommunizieren mit anderen Fahrzeugen (Vehicle-to-Vehicle: V2V), der Infrastruktur, wie z.B. Ampeln (Vehicle-to-Infrastructure: V2I) und mit Services in der Cloud. Der Großteil dieser V2X Kommunikation wird wahrscheinlich IP-basiert sein, wobei im Fall von V2V oder V2I in VANETs auch nicht IP-basierte Kommunikation vorstellbar ist. Für das moderne Fahrzeug ist die V2X Kommunikation von zentraler Bedeutung. Vor allem für eine optimale Navigation und Routen-Planung (bei Elektro-Fahrzeugen z.B. abhängig von der Lade-Infrastruktur), koordiniertes Fahren und die Wartung über "over the air" ECU Software Updates ist V2X Kommunikation unabdingbar. So kann bspw. bei der Navigation und Routen-Planung die aktuelle Verkehrslage berücksichtigt werden, während "over the air" ECU Software Updates die schnelle Wartung einer Vielzahl von Fahrzeugen erlauben, ohne dass diese eine Werkstatt aufsuchen müssen. Auch für die Realisierung autonomer Fahrzeuge spielt V2X Kommunikation eine wichtige Rolle. Die V2X Kommunikation läuft über ein Connectivity-Gateway [26], welches die zentrale Kommunikations-Schnittstelle des Fahrzeugs zur Außenwelt ist. Dafür werden unterschiedliche Technologien wie Wi-Fi (IEEE 802.11), Bluetooth, LTE oder 5G genutzt.

Generell wird im Automobil-Bereich die Kommunikation in 5 Domains aufgeteilt: Antrieb- und Fahrwerksteuerung, Infotainment, Wartung, Sicherheitselektronik (z.B. ABS, Airbag, Gurtstraffer) und Komfort (z.B. elektronische Fensterheber) [23]. V2X Kommunikation umfasst die Domains Infotainment, Wartung und Antrieb- und Fahrwerksteuerung, worunter z.B. Musik-Streams, "over the air" ECU Software Updates und Fahrzeug Kollisionsvermeidung fallen. Einzelne Anwendungsfälle aus der Komfort Domain, wie z.B. das Einstellen der Heizung, könnten ebenfalls realisiert werden. Obwohl V2X Kommunikation generell nicht Echtzeit-kritisch ist, gibt es in machen Fällen, wie der oben genannten Kollisionsvermeidung, harte Echtzeit Deadlines im Millisekunden-Bereich [5], [19], [32].

Es ist also zwingend notwendig, dass die V2X Kommunikation entsprechend abgesichert wird, da sie sicherheitskritische Domains umfasst. Die Absicherung geschieht in erster Linie über ein V2X Security Gateway, welches Teil des Connectivity-Gateways ist (Siehe Abbildung 1). Es besteht

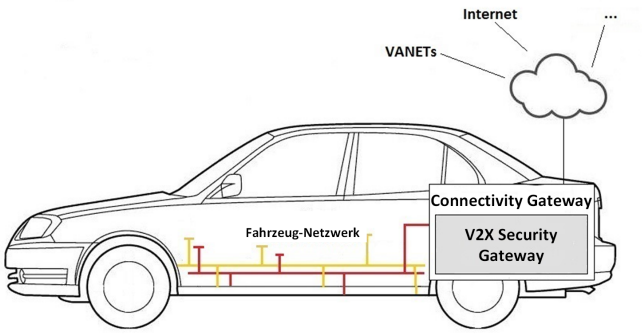

Abbildung 1. Schematisch: Modernes Fahrzeug mit V2X Security Gateway

aus 3 Komponenten, die eine PAP-Struktur (Paket-Filter - Application-Level Gateway - Paket-Filter - Struktur) [8] bilden: 2 zustandsorientierte Paket-Filter, der erste für eingehenden Traffic, der zweite für ausgehenden Traffic und ein V2X Application-Level Gateway (Siehe Abbildung 2, S. 2). Ein Application-Level Gateway (ALG) ist ein Proxy der die Pakete erst nach erfolgreicher Überprüfung auf dem Application-Layer weiterleitet [8]. Sind die Daten verschlüsselt, verfügt das ALG über die nötige kryptografische Funktionalität, um diese zu entschlüsseln und wieder zu verschlüsseln. Solche Security Gateway Lösungen sind ein etabliertes Konzept in der klassischen IT-Security [8], aber in der Automobil IT-Security sind sie eine Neuheit. Die zustandsorientierten Paket-Filter bieten Sicherheit auf dem Internet- und Transport-Layer. Das V2X

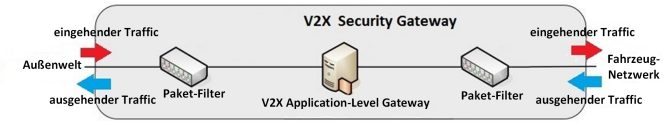

Abbildung 2. Übersicht: V2X Security Gateway Architektur

Application-Level Gateway bietet kryptografische Sicherheit (Vertraulichkeit, Integrität und Authentizität), Sicherheit auf dem Application-Layer, inklusive einer Kontext-sensitiven semantischen Analyse der Anwendungsdaten und dient den Fahrzeug-internen Diensten, die mit der Außenwelt kommunizieren, als Proxy. Dieses allgemeine Konzept des V2X Security Gateways wurde im Grundprojekt [28] ausgearbeitet. Konkret wurden die Anforderungen an ein solches Gateway identifiziert und ausgehend davon die hier vorgestellte Architektur abgeleitet. Die einzelnen Komponenten des V2X Security Gateways wurden dabei nicht entwickelt. Der nächste Schritt ist somit die Entwicklung der zentralen Komponente: des V2X Application-Level Gateways. Zusätzlich zu der genannten Funktionalität soll es den Rollen-basierten Zugriff auf Fahrzeug-interne Ressourcen und eine Bandbreiten-Kontrolle des V2X-Traffics erlauben. Außerdem wird die Auslagerung eines Teils der Funktionalität in die Cloud, wo mehr Ressourcen wie Rechenleistung und Speicher zur Verfügung stehen, diskutiert. Das Ziel dieser Arbeit ist die Ausarbeitung des Konzepts eines solchen V2X Application-Level Gateways. Ausgehend davon wird im Rahmen der Masterarbeit und des Hauptprojekts die V2X Application-Level Gateway Software entwickelt und evaluiert. Die Arbeit am V2X Security Gateway findet im Rahmen der CoRE Gruppe (https://core.informatik.haw-hamburg.de) statt, welche sich aktuell mit der IT-Security im Automobil befasst. Die CoRE Gruppe ist einer der Partner des SecVI (Security for Vehicular Information) Projekts (https://secvi.inet.haw-hamburg.de), dessen Ziel die Entwicklung eines sicheren Fahrzeug-Netzwerks ist. Das V2X Security Gateway ist der Einstiegspunkt in ein solches Fahrzeug-Netzwerk.
Diese Arbeit ist wie folgt aufgebaut: Abschnitt II gibt eine Übersicht über verwandte Arbeiten und stellt den Beitrag dieser Arbeit heraus; in Abschnitt III wird eine Analyse der Anforderungen an ein V2X Application-Level Gateways durchgeführt und das daraus resultierende Konzept vorgestellt und Abschnitt IV fasst diese Arbeit abschließend zusammen und bietet einen Ausblick auf zukünftige Arbeiten.

II. Verwandte Arbeiten

Dieser Abschnitt gibt eine Übersicht über verwandte Arbeiten, welche Security Gateways und Security Proxies behandeln. Da moderne Fahrzeuge durch die zunehmende Vernetzung derselben Entwicklung folgen, wie die Industrie mit dem Konzept der ”Industrie 4.0”, also letztendlich einer Integration in das IoT, umfassen die Arbeiten neben dem Automobil-Bereich auch den Industrie-Bereich, das IoT und den traditionellen Web-Bereich.

A. Automotive Security Gateways und Proxys

Die meisten Arbeiten zu Security Gateways aus dem Automobil-Bereich behandeln ausschließlich Gateways zur Absicherung der Kommunikation des internen Fahrzeug-Netzwerks , wie die Arbeit von Pese et al. [22]. In [22] stellen Pese et al. ihr Konzept und eine Prototyp-Implementierung einer Automotive Firewall zum Verhindern von Angriffen zwischen Domains in einem Ethernet-basierten Fahrzeug-Netzwerk mit Domain-Architektur vor. In einer Domain-Architektur wird das Netzwerk in mehrere Segmente aufgeteilt, wobei jedes Segment einer Domain aus dem Automobil-Bereich wie z.B. Antrieb- und Fahrwerksteuerung, Infotainment usw. entspricht und über einen Domain Controller mit dem restlichen Netzwerk verbunden ist. Die Domain Controller sind über einen Ethernet-Backbone miteinander verbunden. Die Automotive Firewall soll Angriffe zwischen den Domains verhindern, also Angriffe aus einer Domain heraus auf Devices einer anderen Domain. Sie besteht aus einem zustandslosen Paket-Filter, welcher Ethernet-Traffic nach konfigurierten Regeln filtert und einem zustandsorientierten Paket-Filter, welcher TCP/IP- und UDP-Traffic nach konfigurierten Regeln filtert. Der zustandslose Paket-Filter ist aus Performance Gründen in Hardware implementiert, während der komplexere zustandsorientierte Paket-Filter in Software implementiert ist. Der zustandslose Paket-Filter ist zwischen den Segmenten des Netzwerks verbaut, um den kompletten Ethernet-Traffic zwischen den Domains filtern zu können. Der zustandsorientierte Paket-Filter wird auf jedem Domain Controller implementiert und filtert den einkommenden Traffic, welcher den zustandslosen Paket-Filter passiert hat. Die Automotive Firewall bietet Sicherheit auf dem Link- und Transport-Layer. In einem zustandslosen Paket-Filter kann bei Bedarf auch das Filtern von IP-Traffic nach konfigurierten Regeln implementiert werden, sodass eine solche Automotive Firewall auch Sicherheit auf dem Internet-Layer bietet.
Von den Arbeiten, die sich mit V2X Sicherheit beschäftigen, fokussieren sich die meisten auf die Realisierung der kryptografischen Absicherung von V2X Kommunikation [30], z.B. im Kontext von ECU Software Updates [15] oder auf Sicherheitsfragen in VANETs, wie Authentifizierung [10], [14], Denial-of-Service (DoS) Angriffe [4] oder Misbehavior Detection [24], [13]. Unter Misbehavior Detection versteht man die Erkennung von Netzwerk-Knoten, welche absichtlich oder aufgrund einer Fehlfunktion falsche Informationen an das Netzwerk senden, z.B. im VANET Kontext fälschlicherweise einen Unfall melden. Die wenigsten behandeln die Absicherung der V2X Kommunikation über eine im Fahrzeug verbaute Komponente, wie ein V2X Security Gateway oder Proxy, deren Funktionalität über die reine Ver- und Entschlüsselung und Prüfung von Signaturen und Zertifikaten hinausgeht. Eine dieser Arbeiten ist die von Bouard et al. [6], welche einen Security Proxy vorstellt. Sie behandelt die Absicherung der Kommunikation zwischen (Consumer Electronics) Devices und ECUs des internen Fahrzeug-Netzwerks über einen Security Proxy. Der Security Proxy entkoppelt die CE Devices von den ECUs und stellt sicher, dass nur autorisierte Devices mit den ECUs kommunizieren. Der Security Proxy kommuniziert

über eine sichere, IP-basierte Middleware, z.B. SEIS [12], mit den ECUs und leitet nur Nachrichten von authentifizierten CE Devices an sie weiter.
Der Beitrag dieser Arbeit ist das Konzept eines V2X Application-Level Gateways zur Absicherung der V2X Kommunikation. Im Gegensatz zu [6] beschränkt sich das V2X Application-Level Gateway nicht auf die Kommunikation mit CE Devices und bietet mehr Funktionalität als die bisher vorgestellten V2X Security Proxys und Application-Level Gateways. Neben der kryptografischen Funktionalität und der Funktion des Proxys für Fahrzeug-interne Dienste die mit der Außenwelt kommunizieren, sichert es die Kommunikation auf dem Application-Layer, was eine semantische Analyse der Anwendungsdaten beinhaltet, verhindert durch Rollen-basierten Zugriff auf Fahrzeug-interne Ressourcen den Zugriff auf diese durch Unberechtigte über V2X und erlaubt eine Bandbreiten-Kontrolle des V2X-Traffics. Bei der Absicherung der V2X Kommunikation ergänzt das V2X Application-Level Gateway Paket-Filter Lösungen, welche den Internet- und Transport-Layer absichern, wie z.B. [22]. Zusammen bilden sie ein V2X Security Gateway für einen umfassenden Schutz der V2X Kommunikation.

B. IoT und Industrie Security Gateways und Proxys

Im IoT und im Industrie-Bereich lag der Fokus bei Gateways und Proxys häufig auf der Übersetzung von Protokollen zur Gewährleistung der Interoperabilität von heterogenen Devices oder Systemen. Zunehmend rückt nun auch die Sicherheit in den Fokus. Ein Beispiel ist das in [21] vorgestellte IoT Gateway, welches TLS für die kryptografische Sicherheit nutzt. Durch den Einsatz von TLS werden die Daten verschlüsselt und die Authentifizierung der Kommunikationspartner ermöglicht. Der in [9] vorgestellte IoT Security Proxy benutzt symmetrische Verschlüsselungs-Algorithmen für die Verschlüsselung von Daten, die zwischen dem Proxy und Devices ausgetauscht werden und für die Authentifizierung von Devices, womit die kryptografische Sicherheit angegangen wird. Die symmetrischen Verschlüsselungs-Algorithmen wurden gewählt, weil die einfach und performant sind. Außerdem werden Access Control Lists (ACLs) verwendet, welche die Operationen festlegen, die jede User-Gruppe ausführen darf. Um eine durch eine ACL geschützte Operation ausführen zu können, z.B. den Zugriff auf eine geschützte Datei, muss der Anfrage ein Zertifikat hinzugefügt werden, welches beweist, dass der Anfragende zu einer User-Gruppe gehört, welche die Operation ausführen darf. Der in [29] beschriebene Industrie Security Proxy beschränkt den Zugriff auf Ressourcen, z.B. die Ausführung einer Operation, durch einen Rollen-basierten Zugriff auf diese. Außerdem wird eine sehr einfache Form der semantischen Analyse von Anwendungsdaten durchgeführt: wenn ein Befehl über den Security Proxy an ein Device gesendet wird, überprüft der Security Proxy, ob der Befehl in der Menge der Befehle ist, welche das Device ausführen kann und leitet ihn nur dann weiter. Der in [31] beschriebene Proxy erlaubt eine Bandbreiten-Kontrolle. Dafür werden alle Netzwerk-Streams in einer so genannten Stream-Hierarchie eingeordnet, welche durch einen Graphen repräsentiert wird. Die Blatt-Knoten dieses Graphen repräsentieren die Netzwerk-Streams. Die inneren Knoten implementieren eine bestimmte Aufteilung der Bandbreite auf ihre Kind-Knoten. Die Aufteilung ”Mutex” bspw. stellt sicher, dass immer höchstens ein Kind-Knoten Bandbreite zugewiesen bekommt. Weitere Aufteilungen sind ”Priority”, welche Bandbreite anhand von Prioritäten der Kind-Knoten zuweist und ”Weight”, welche die verfügbare Bandbreite auf alle Kind-Knoten anhand definierter Gewichtungen der Netzwerk-Streams aufteilt. Das in [20] vorgestellte IoT Gateway ”LEGIoT” benutzt Docker (https://www.docker.com/) für die Virtualisierung all seiner Komponenten über Docker Container, was einen schnellen Building Prozess und Instanziierung und ein einfaches Management erlaubt. Außerdem wird durch die Isolierung von Komponenten eine höhere Flexibilität des Systems erreicht. Der Nachteil von Virtualisierung ist der erhöhte Bedarf an Ressourcen, z.B. Speicher und CPU Leistung, im Fall von [20] um die Docker Engine, welche die Virtualisierung auf dem System realisiert, auszuführen. Die hier aufgezählte Funktionalität aus dem IoT und Industrie-Bereich lässt sich in ein V2X Application-Level Gateway übertragen. Zusammengefasst, sind das zusätzlich zur Proxy-Funktionalität, kryptografischen Sicherheit und Sicherheit auf dem Application-Layer: ACLs für einen Rollen-basierten Zugriff auf Ressourcen, Bandbreiten-Kontrolle und Virtualisierung.

C. Web Security Gateways und Proxys

Allgemein bieten Web Security Gateways und Web Security Proxys ähnliche Funktionalität wie IoT oder Industrie Security Gateways und Proxys: Proxy-Funktionalität, Verschlüsselung, Application-Layer Sicherheit [17], [7], welche in diesem Fall auf Web Protokolle wie HTTP beschränkt ist [18], Bandbreiten-Kontrolle [3] und die semantische Analyse von Anwendungsdaten [27]. Der in [27] vorgestellte Web Security Proxy analysiert Anwendungsdaten und klassifiziert sie anhand vordefinierter Regeln, z.B. ”der Datentyp muss *int* sein”, entweder als gültig oder ungültig. Der Regelsatz kann problemlos erweitert werden. Nur wenn die Daten gültig sind, wird der entsprechende HTTP Request/Response weitergeleitet. Ein zusätzliches nützliches Feature von Web Security Gateways und Proxys ist Logging [18]. Da moderne Fahrzeuge sowohl Consumer als auch Providervon Web Services sind, kann die hier aufgezählte Funktionalität in ein V2X Application-Level Gateway übertragen werden.

III. KONZEPT

In diesem Abschnitt wird das Konzept des V2X Application-Level Gateways vorgestellt, beginnend mit einer Analyse der Anforderungen, gefolgt von der daraus abgeleiteten Architektur und einer Beschreibung des Aspekts der semantischen Analyse. Der Abschnitt wird mit einer Diskussion eines teilweise Cloud-basierten Ansatzes des V2X Application-Level Gateways abgeschlossen.

A. Anforderungen

Die Anforderungen werden in funktionale- und Performance-Anforderungen unterteilt. Für das V2X

Application-Level Gateway ergeben sich diese aus seiner Aufgabe die V2X Kommunikation abzusichern. Es wurden die folgenden Anforderungen identifiziert:

Funktionale Anforderungen:

1) Gewährleistung der kryptografischen Sicherheit: Sicherstellung der Vertraulichkeit, Integrität und Authentizität der V2X Kommunikation, um Fahrzeug- und Fahrer-bezogene Daten zu schützen und das Fahrzeug selbst gegen Angriffe und Manipulation absichern. Für die Kommunikation zwischen dem V2X Application-Level Gateway und der Außenwelt muss eine starke Verschlüsselung verwendet werden, während für die Kommunikation zwischen dem V2X Application-Level Gateway und Fahrzeug-internen Diensten eine schwächere Verschlüsselung verwendet werden kann, um die ECUs zu entlasten.

2) Absicherung auf dem Application-Layer: Kontrolle der Daten auf dem Application-Layer nach vordefinierten Regeln, inklusive einer Kontext-sensitiven semantischen Analyse (Siehe Absatz III-C), was erfordert, dass das V2X Application-Level Gateway alle Fahrzeug-internen Dienste, die V2X nutzen, kennt, um die entsprechenden Sicherheits-Konfigurationen (welche die Regeln etc. enthalten) zu laden. Dies verbessert den Schutz der Fahrzeug- und Fahrer-bezogenen Daten und des Fahrzeugs vor Angriffen und Manipulation.

3) Proxy-Funktionalität: dient den Fahrzeug-internen Diensten als Proxy für die Kommunikation mit der Außenwelt, um das Fahrzeug-Netzwerk von externen Kommunikationspartnern zu entkoppeln.

4) Realisierung eines Rollen-basierten Zugriffs auf Ressourcen über ACLs, um unautorisierten Zugriff zu verhindern.

5) Bandbreiten-Kontrolle des V2X-Traffics: Aufteilung der Bandbreite unter den Anwendungen (wie in [31]), z.B. bei Bedarf eine Priorisierung sicherheitskritischer Dienste, um die Performance des Fahrzeugs in jeder Situation zu optimieren.

6) Unterstützung IP-basierter Anwendungsprotokolle (z.B. HTTP), da ein Großteil der V2X Kommunikation wahrscheinlich IP-basiert sein wird.

7) Konfigurierbarkeit des Systems, z.B. Update oder Hinzufügen von neuen Regeln für die semantische Analyse oder Bandbreiten-Kontrolle, um die Wartung zu erleichtern und eine optimale Performance über einen langen Produkt-Lebenszyklus zu ermöglichen.

8) Logging-Funktionalität zur Erleichterung der Wartung, was Meldungen an die Cloud einschließt.

Performance Anforderungen:

1) Echtzeitfähigkeit mit Deadlines im Millisekunden-Bereich ($<$10 ms) im Fall von Echtzeit-kritischer V2X Kommunikation wie Kollisionsvermeidung. Für nicht Echtzeit-kritischen V2X-Traffic liegt die Ende-zu-Ende Latenz im Bereich 100 ms bis $>$1s [5], [19], [32].

2) Ausreichend Durchsatz (der Durchsatz für V2X Anwendungen liegt im Bereich von 5 Kbps bis 700 Mbps und beträgt für einen Großteil der Anwendungen 10 bis 80 Mbps [5], [19]) für jeglichen V2X-Traffic.

B. Architektur

Die Architektur des V2X Application-Level Gateways wurde ausgehend von den Anforderungen und dem Konzept der Service-orientierten Kommunikation auf Grundlage der in [25] beschriebenen allgemeinen Best-Practice Application-Level Gateway Software Architektur entwickelt. Die Architektur von [25] (Siehe Abbildung 3, S. 5) setzt mehrere Design Pattern um und ist erweiterbar. Der Input wird vom Output entkoppelt, d.h. es gibt separate Komponenten für die Verarbeitung von eingehenden und ausgehenden Daten (Router Pattern). Die Service-Initialisierung wird von den Funktionen, welche der Dienst ausführt nachdem er initialisiert worden ist, entkoppelt, d.h. es gibt separate Komponenten für die Initialisierung (z.B. Verbindungsaufbau etc.) von Diensten (Acceptor und Connector Pattern). Im Fall eines Application-Level Gateways sind das Dienste zur Verarbeitung der eingehenden und ausgehenden Daten (Input und Output). Außerdem werden Events (z.B. Connection-Request) von einem zentralen Reactor an die entsprechenden Event-Handler (z.B. Acceptor) verteilt (Reactor Pattern). Connection-Requests oder Daten, im V2X Kontext entweder von Fahrzeug-internen Diensten oder externen Diensten, werden von den *Communication Endpoints* empfangen. Im Fall eines Connection-Requests (in Abbildung 3: ”1. Connection Request”, grün) von einem Dienst (Service Consumer) benachrichtigt der *Reactor* den *Acceptor* (”2. Notify”, grün), welcher dann eine Verbindung zwischen dem Dienst und dem Application-Level Gateway herstellt (”3. Connection Request”, grün). Der *Connector* wird verwendet, um aktiv Verbindungen vom Application-Level Gateway zu Diensten (Service Providers) aufzubauen (in Abbildung 3: ”4. Connection Request”, grün). Im Fall von eingehenden Daten (in Abbildung 3: ”1. Data”, rot) benachrichtigt der *Reactor* den *Input Channel* (”2. Notify”, rot), welcher dann die Daten erhält (”3. Data”, rot) und die *Routing Table* abfragt (”4. Get Destination”, rot), um die Daten über den *Output Channel* (”5. Data”, rot) an das Ziel weiterzuleiten (”6. Data”, rot. Es kann auch mehrere *Input Channel* und *Output Channel* geben. Der *Input Channel* und *Output Channel* realisieren die Proxy-Funktionalität des Application-Level Gateways, womit die Architektur die funktionale Anforderung ”*3)*” erfüllt.
Diese Architektur wurde um mehrere Komponenten erweitert, um alle in Abschnitt III-A definierten Anforderungen zu erfüllen (Siehe Abbildung 4, S. 5). Die zusätzlichen Komponenten (in Abbildung 4 grau) realisieren die Security-Funktionalität, während die Komponenten der Grundarchitek-

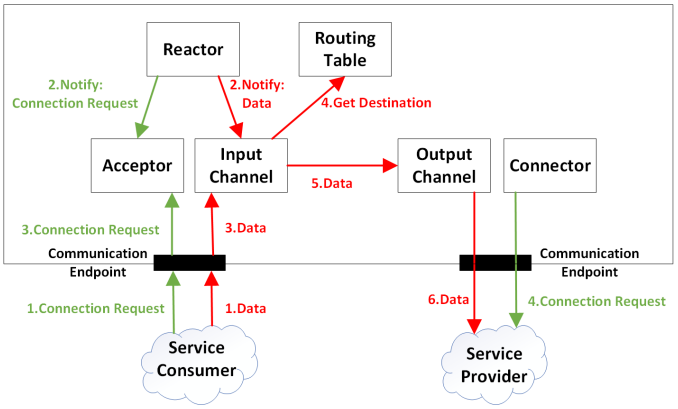

Abbildung 3. Übersicht: Best-Practice Application-Level Gateway Software Architektur

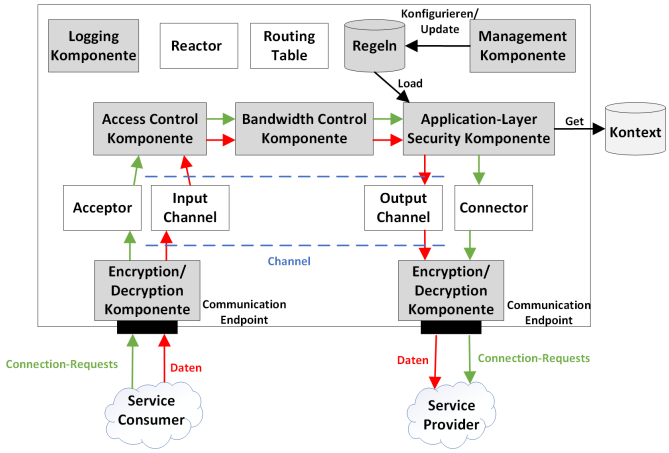

Abbildung 4. Übersicht: V2X Application-Level Gateway Architektur

tur (in Abbildung 4 weiß) die grundlegende Kommunikations-Funktionalität realisieren. Die *Encryption/Decryption Komponenten* für eingehenden und ausgehenden Traffic bieten kryptografische Sicherheit, womit Anforderung *"1)"* erfüllt wird. Aus Performance-Gründen muss die Funktionalität für die Gewährleistung der kryptografischen Sicherheit in speziell dafür konzipierte Hardware-Bausteine ausgelagert werden [30], wie z.B. EVITA HSMs (Hardware Security Module) [1]. Die *Encryption/Decryption Komponenten* gehören somit zwar zum V2X Application-Level Gateway als Ganzem, sind aber nicht Teil der Software des V2X Application-Level Gateways, welche auf einem eigenen Device läuft. Die *Access Control Komponente* verwaltet die ACLs, welche den Rollen-basierten Zugriff auf Ressourcen realisieren, womit Anforderung *"4)"* erfüllt wird. Eine Nachricht wird nur dann weitergeleitet, wenn sie laut den ACLs gültig ist. Die *Bandwidth Control Komponente* erlaubt eine Kontrolle der Bandbreite des V2X-Traffics gemäß der in Abschnitt II-B beschriebenen Verteilungsmechanismen, womit Anforderung *"5)"* erfüllt wird. Die *Application-Layer Security Komponente* überprüft die Anwendungsdaten gemäß konfigurierbarer Regeln, was bei Bedarf eine Kontext-sensitive semantische Analyse einschließt, womit Anforderung *"2)"* erfüllt wird. Sie muss alle relevanten IP-basierten Protokolle unterstützen, was Anforderung *"6)"* erfüllt. Eine Nachricht wird nur dann weitergeleitet, wenn sie als semantisch korrekt eingestuft wird (Siehe Abschnitt III-C). Mit der *Logging Komponente*, welche die Aktivitäten aller Komponenten aufzeichnet, wird Anforderung *"8)"* erfüllt. Über die *Management Komponente* kann die *Application-Layer Security Komponente*, *Access Control Komponente*, *Bandwidth Control Komponente* und die *Logging Komponente* konfiguriert werden, womit Anforderung *"7)"* erfüllt wird. Die entwickelte Architektur erfüllt somit alle in Abschnitt III-A definierten funktionalen Anforderungen.

Jede Nachricht wird sequentiell von einer festen Anzahl an Komponenten verarbeitet. Eine parallele Verarbeitung von Nachrichten durch das Vorhandensein mehrerer paralleler Channels (ein Channel besteht aus einer *Input Channel*- und *Output Channel*-Komponente, in Abbildung 4 durch die gestrichelten blauen Linien hervorgehoben), z.B. separate Channels für Echtzeit-kritische Daten, ist möglich. Im Falle mehrerer paralleler Channels können bei Bedarf auch mehrere parallel laufende Instanzen der *Access Control*- und *Application-Layer Security Komponente* vorhanden sein (z.B jeweils eine für jeden Channel). Abhängig von der Konfiguration der Kontrolle der Bandbreite können gegebenenfalls auch mehrere parallel laufende Instanzen der *Bandwidth Control Komponente* vorhanden sein. Somit ermöglicht die Architektur prinzipiell auch die Erfüllung aller in Abschnitt III-A definierten Performance-Anforderungen.

Das V2X Application-Level Gateway muss alle Fahrzeug-internen Dienste, welche V2X verwenden, kennen. Das kann mit einer zentralen Fahrzeug Service-Registry realisiert werden, welche diese Dienste enthält. Die allgemeine Registrierung und das Finden von Diensten stehen nicht im Fokus dieser Arbeit , weshalb sie nur kurz im Kontext des Updates der Liste von V2X Diensten des V2X Application-Level Gateways beschrieben werden (Siehe Abbildung 5). Jeder Fahrzeug-interne V2X Service-Provider muss sich in einer (Fahrzeug) Service-Registry registrieren (in Abbildung 5: "1. Register", rot). Ein externer Service-Consumer kann diesen Service-Provider finden ("2. Look up", rot) und dann können sie über das V2X Application-Level Gateway kommunizieren ("3. Kommunikation", rot). Umgekehrt registriert sich ein

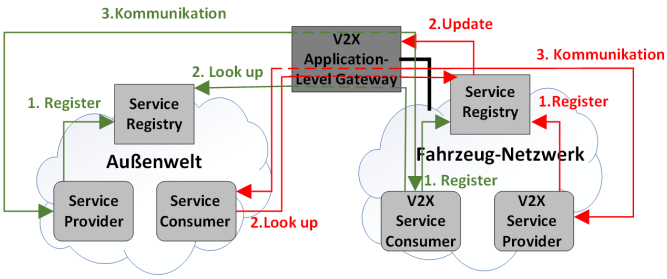

Abbildung 5. Kommunikation zwischen Fahrzeug-internen und externen Diensten

externer Service-Provider bei einer Service-Registry (in Abbildung 5: "1. Register", grün), sodass ein Fahrzeug interner V2X Service-Consumer diesen finden ("2. Look up", grün) und mit ihm kommunizieren kann ("3. Kommunikation", grün). Zusätzlich muss sich der Fahrzeug interne V2X Service-

Consumer bei der Fahrzeug Service-Registry registrieren (in Abbildung 5: ”1. Register”, grün), damit bei jeder Registrierung eines Fahrzeug-internen V2X Service-Providers oder Consumers die Liste von V2X Diensten des V2X Application-Level Gateways geupdatet werden kann. So kann für jeden registrierten Dienst immer die entsprechende Security-Konfiguration geladen werden, welche die Regeln zur Prüfung der Daten etc. enthält.

C. Semantische Analyse

In dieser Arbeit wird die semantische Analyse von Daten als die Überprüfung der Konformität syntaktisch korrekter Daten mit einer Menge semantischer Regeln definiert. Die Daten werden als semantisch korrekt eingestuft, wenn sie Regel-konform sind. Eine Regel fällt in eine der beiden allgemeinen Klassen: strukturelle Regeln und inhaltliche Regeln. Strukturelle Regeln beziehen sich auf Eigenschaften wie Datentyp oder Payload-Größe, während inhaltliche Regeln sich auf die anwendungsbezogene Bedeutung der Daten beziehen. Eine Regel kann außerdem Kontext-abhängig oder Kontext-unabhängig sein. Der String ”Hamburg” ist bswp. ein semantisch korrektes Ziel für ein Navigationssystem, während es der String ”Asdf” nicht ist. In diesem Fall ist die inhaltliche Regel, dass der String in einer definierten Menge bekannter Ziele ist. Ein anderes Beispiel ist eine Nachricht, welche den Teil eines Software Updates enthält. Eine strukturelle Regel für eine solche Nachricht ist, dass die Größe (in Bytes) ihrer Payload in einem bestimmten Bereich liegt (für den letzten Teil des Updates kann dieser Bereich abweichend sein, da der letzte Teil des Updates möglicherweise nur einige wenige Bytes groß ist). Ergibt nun die Überprüfung der Größe der Payload, dass diese nur einige wenige Bytes groß ist und es sich bei der Nachricht nicht um die letzte Nachricht handelt, wird sie als ungültig eingestuft. Das System reagiert dann entsprechend, indem die Nachricht nicht weitergeleitet wird, sondern z.B. einfach gedroppt wird. In den eben genannten Beispielen ist die semantische Korrektheit unabhängig vom Kontext und die semantische Analyse kann unabhängig vom Fahrzeug-Systemzustand durchgeführt werden. In vielen Fällen hängt die semantische Korrektheit aber gerade vom Systemzustand ab und für die semantische Analyse ist Kontext-Sensitivität nötig. Abbildung 6 enthält State-Machines, welche das korrekte Verhalten des Kofferraums und des Motors eines Fahrzeugs beschreiben. Für eine bessere Übersicht sind nur die gültigen Zustandsübergänge ohne Selbst-Transitionen abgebildet. Der Befehl den Kofferraum abzuschließen (”Lock”) ist in der Menge der bekannten Befehle enthalten, aber semantisch inkorrekt, wenn der Kofferraum geöffnet ist (Zustand ”OPEN”). Für eine semantische Analyse reicht es in manchen Fällen nicht aus, den Zustand nur einer Komponente, z.B. des Kofferraums, zu kennen. Stattdessen muss der Zustand einer Komposition von Komponenten, z.B. Kofferraum und Motor, bekannt sein. Der Befehl den Kofferraum zu öffnen (”Open”) ist, unabhängig vom Zustand des Kofferraums, semantisch inkorrekt, wenn das Fahrzeug gerade fährt (Motor-Zustand ”DRIVING”).
Das V2X Application-Level Gateway muss die aktuellen Zustände aller für seine Überprüfungen relevanten Komponenten kennen. Das kann auf zwei unterschiedliche Weisen realisiert werden: entweder wird das V2X Application-Level Gateway bei Zustandsänderungen benachrichtigt (”Notification”), oder es fragt die aktuellen Zustände ab (”Request”). Der Vorteil des Notification-Ansatzes ist, dass es keine unnötige Kommunikation gibt: Nachrichten werden nur gesendet und empfangen, wenn eine Zustandsänderung stattfindet. Der Nachteil ist, dass sich das V2X Application-Level Gateway darauf verlassen muss, dass die Notification durch ein anderes Device auch in jedem Fall stattfindet. Bei dem Request-Ansatz gibt es diese Abhängigkeit von der Zuverlässigkeit (hinsichtlich des Versendens) anderer Devices nicht, aber es gibt Netzwerk-Traffic auch wenn keine Zustandsänderung stattfindet. Finden Zustandsänderungen im Vergleich zu Requests verhältnismäßig selten statt, bedeutet das unnötigen Netzwerk-Traffic. Es handelt sich hierbei aber um keine großen Datenmengen, weshalb dieser Traffic kein Problem darstellt. Das V2X Application-Level Gateway kann die aktuellen Zustände entweder periodisch abfragen, oder wenn ein entsprechendes Ereignis stattfindet (”Event-driven”). Das Eintreffen von Daten zur Analyse ist ein mögliches solches Ereignis.
Ein generelles Problem bei der Kontext-Sensitivität des V2X Application-Level Gateways ist die temporäre Inkonsistenz zwischen dem Systemzustand aus der Sicht des Gateways und dem tatsächlichen Systemzustand, aufgrund des Propagation Delays der Zustandsänderung. Bei dem Notification-Ansatz gibt es eine Inkonsistenz bei jeder Zustandsänderung (Siehe Abbildung 7). Bei einer Zustandsänderung, z.B. zur Zeit *t2*

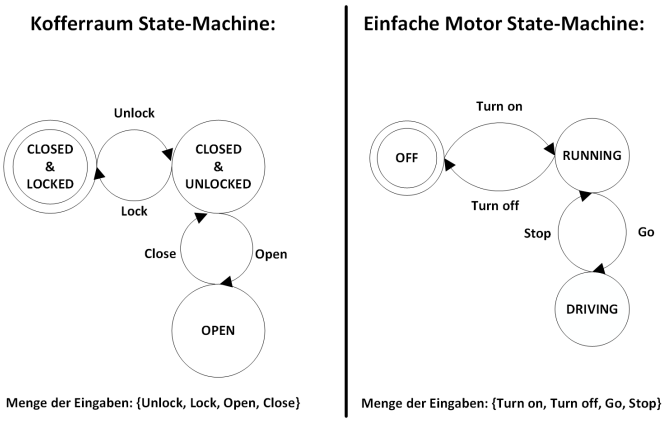

Abbildung 6. Einfache State-Machines zur Beschreibung des korrekten Verhaltens des Kofferraums und Motors eines Fahrzeugs

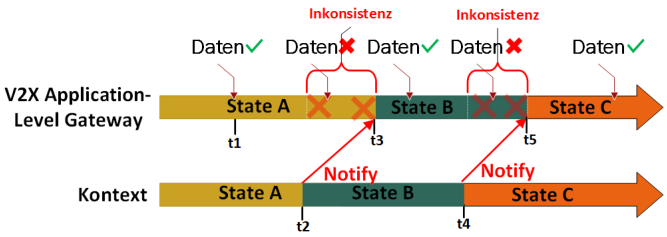

Abbildung 7. Inkonsistenz des Systemzustands bei Notification

oder *t4*, liegt eine Inkonsistenz vor, bis die Benachrichtigung das V2X Application-Level Gateway erreicht (jeweils Zeitpunkte *t3* und *t5*). Daten, die zwischen *t2* und *t3* oder *t4* und *t5* eingehen, können somit nicht korrekt analysiert werden, da zur Zeit der Analyse nicht der korrekte Systemzustand vorliegt.
Wenn der aktuelle Zustand periodisch abgefragt wird, liegt ebenfalls eine Inkonsistenz bei jeder Zustandsänderung vor (Siehe Abbildung 8), bis die Antwort (Zeitpunkt *t4*) des periodischen Requests (Zeitpunkt *t3*) das V2X Application-Level Gateway erreicht (Zeitpunkt *t5*). Daten, die zwischen *t2* und *t5* eingehen, können somit nicht korrekt analysiert werden.

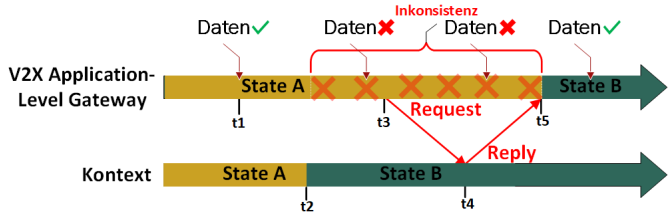

Abbildung 8. Inkonsistenz des Systemzustands bei periodischen Requests

Wird der aktuelle Zustand ”Event-driven”, z.B. beim Eingehen von Daten zur Analyse, abgefragt, liegt nur dann eine Inkonsistenz vor, wenn der Zustand sich verändert, nachdem der Request bearbeitet wurde, aber bevor die Antwort des Requests das V2X Application-Level Gateway erreicht (Siehe Abbildung 9). Der Request wird zur Zeit *t2* verarbeitet und die zugehörige Antwort erreicht das V2X Application-Level Gateway zum Zeitpunkt *t4*. Aber im Zeitpunkt *t3* findet eine Zustandsänderung statt, was eine Inkonsistenz zur Folge hat. Die Daten, die zur Zeit *t1* zur Analyse beim V2X Application-Level Gateway eingingen, können somit nicht korrekt analysiert werden. Erst neue Daten, die zum Zeitpunkt *t5* eingehen und den nächsten Request auslösen, können mit der zur Zeit *t7* beim V2X Application-Level Gateway eintreffenden Antwort wieder korrekt analysiert werden.

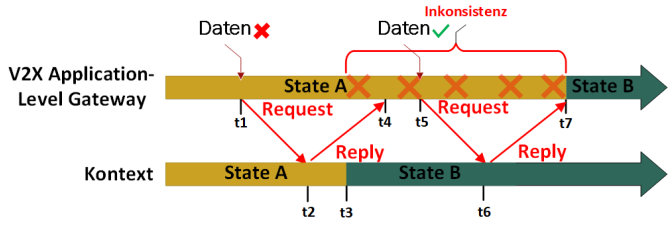

Abbildung 9. Inkonsistenz des Systemzustands bei Requests ”on demand”

In der Praxis findet eine Zustandsänderung des Kontexts in der Regel nicht in einem diskreten Zeitpunkt statt, sondern dauert eine gewisse Zeit (t_{dS}), wie z.B. das Öffnen oder Schließen eines Kofferraums. Nach einer Zustandsänderung bleibt der Zustand in der Regel auch für eine gewisse Zeit ($t_{s\ const}$) konstant, bevor eine weitere Änderung möglich ist. Unter diesen Annahmen lässt sich das Inkonsistenz-Problem mit dem Notification-Ansatz umgehen, wenn die Signallaufzeit der Notification ($t_{\ notify}$) ausreichend kurz ist (Siehe Abbildung 10). Zu Beginn einer Zustandsänderung (*t1*) wird eine Notification mit dem bekannten Zeitpunkt der Vollendung dieser Zustandsänderung (*t3*) an das V2X Application-Level Gateway gesendet. Erreicht diese mit einem Delay von $t_{\ notify}$ das V2X Application-Level Gateway (*t2*), kann es den Zustand zum Zeitpunkt *t3* mit dem Kontext synchronisieren. Auf diese Weise wird eine temporäre Inkonsistenz vermieden und es liegt für jede Analyse der korrekte Systemzustand vor. Für diesen Ansatz ist eine Synchronisation der Uhren des V2X Application-Level Gateways und der Komponente, welche den Kontext hält, notwendig. Ist die erfolgreiche Vollendung einer Zustandsänderung nicht immer garantiert, wird für die Vermeidung einer Inkonsistenz eine zweite Notification benötigt (*t3*), welche dem V2X Application-Level Gateway die erfolgreiche Vollendung bestätigt (*t4*). Die Analyse von Daten, welche nach einer erwarteten Zustandsänderung (*t3*), aber vor der Bestätigung der erfolgreichen Vollendung dieser Zustandsänderung (*t4*) eintreffen, muss bis zum Eintreffen der Bestätigung verzögert werden.

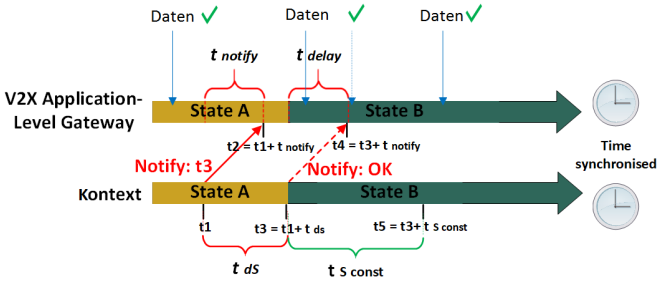

Abbildung 10. Inkonsistenz bei nicht zeitdiskreten Zustandsänderungen

In jeder Fahrzeug-Netzwerk Sicherheits-Architektur ist das V2X Security Gateway nur die erste Linie der Verteidigung. Deshalb kann eine semantische Analyse der eingehenden Daten, z.B. Befehl zum Öffnen des Kofferraums (”Open trunk”), nicht nur vom V2X Application-Level Gateway, sondern zusätzlich auch von einer ECU im Fahrzeug-Netzwerk, z.B. einem Domain Controller, durchgeführt werden (Siehe Abbildung 11). Voraussetzung dafür ist, dass die ECU die Daten entschlüsseln kann. Wenn eine semantische Analyse derselben Daten in mehreren Komponenten durchgeführt wird, sollte sie optimal verteilt werden. Für maximale Sicherheit sollte es möglichst viel Redundanz bei der Analyse in mehreren Komponenten geben. So kann der Ausfall einer kompromittierten Komponente kompensiert werden. Allerdings muss auch der Einfluss der Analyse auf die Performance der Komponenten beachtet werden. Muss bspw. das V2X Application-Level Gateway bereits viel Traffic verarbeiten, kann hier die semantische Analyse aus Performance-Gründen auf eine allgemeine Analyse auf höherem Abstraktions-Level beschränkt werden, z.B. Untersuchung des Datentyps. Eine spezifischere, z.B. Kontext-sensitive Analyse würde anschließend bspw. im jeweils zuständigen Domain Controller durchgeführt werden. Aus Performance-Gründen kann also auch auf eine ”Verzahnung” gesetzt werden bei der wenig (oder keine) redundante Analyse durchgeführt wird, aber die Analyse beider Komponenten in Summe die vollständige Semantik der Daten abdeckt.

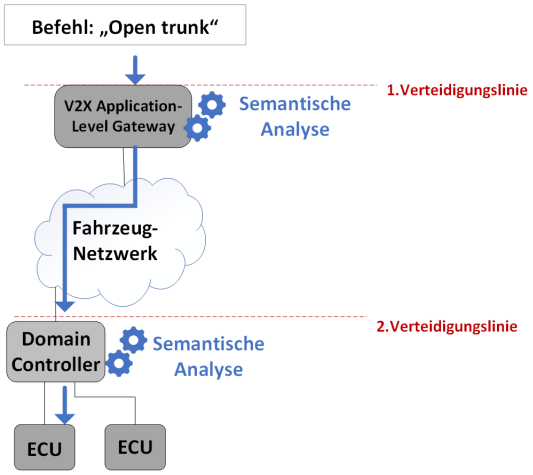

Abbildung 11. Einfaches Beispiel für verteilte semantische Analyse im Fahrzeug

D. Cloud-basierter Ansatz

Das V2X Application-Level Gateway muss nicht notwendigerweise als eine Komponente im Fahrzeug verbaut werden. Stattdessen kann ein Teil der Funktionalität in die Cloud ausgelagert werden (Siehe Abbildung 12, S. 9), um die im Fahrzeug verbaute V2X Application-Level Gateway Komponente zu entlasten. Um die benötigte geringe Latenz der direkten V2V und V2I VANET-Kommunikation zu erhalten, wird diese weiterhin vollständig von der im Fahrzeug verbauten Komponente verarbeitet. Aber jegliche Cloud-basierte V2X Kommunikation könnte von der V2X Application-Level Gateway Komponente in der Cloud verarbeitet werden. Die im Fahrzeug verbaute Komponente würde nur über eine sichere Verbindung mit der Cloud-Komponente kommunizieren und sich bei Cloud-basierter Kommunikation nur auf die Verschlüsselung und das Mapping von externen Adressen auf Fahrzeuginterne Adressen beschränken. Die restliche Funktionalität des V2X Application-Level Gateways, wie Application-Layer Sicherheit inklusive semantische Analyse oder Realisierung des Rollen-basierten Zugriffs auf Ressourcen würde in die Cloud ausgelagert werden, wo deutlich mehr Ressourcen wie Rechenleistung und Speicher zur Verfügung stehen, als im Fahrzeug. Ein weiterer Vorteil einer solchen Auslagerung ist der ökonomische Aspekt. In Fahrzeugen könnten leistungsschwächere und günstigere V2X Application-Level Gateway Komponenten verbaut werden, während in der Cloud viele V2X Application-Level Gateway Komponenten auf einem Server laufen könnten. Ein Nachteil dieser Lösung ist die Abhängigkeit des Fahrzeugs von einer zuverlässigen Cloud-Infrastruktur.

IV. Zusammenfassung und Ausblick

In dieser Arbeit wurde das Konzept eines V2X Application-Level Gateways entwickelt. Die wesentlichen Anforderungen an ein solches Gateway sind, dass es die kryptografische Sicherheit der V2X Kommunikation gewährleistet, die Kommunikation auf dem Application-Layer Sicherheit absichert und Proxy-Funktionalität bietet. Für die Application-Layer Sicherheit ist eine semantische Analyse der Anwendungsdaten nötig, was in manchen Fällen Kontext-Sensitivität erfordert. Zusätzlich erlaubt das Gateway eine Bandbreiten-Kontrolle des V2X-Traffics und Rollen-basierten Zugriff auf Ressourcen im Fahrzeug über ACLs. Ausgehend von diesen Anforderungen wurde die Architektur des V2X Application-Level Gateways, basierend auf der Architektur für Application-Level Gateways von [25], entwickelt. Mit dieser Architektur können die genannten Anforderungen erfüllt werden.

Das Ziel der weiteren Arbeit im Rahmen des SecVI Projekts in der CoRE Gruppe ist die Implementierung eines Prototypen des V2X Application-Level Gateways. Dieser wird mit einem Test-Aufbau evaluiert, bei dem die V2X Application-Level Gateway Software auf einem Linux PC läuft, während das interne Fahrzeug-Netzwerk und die mit dem Fahrzeug kommunizierenden externen Dienste jeweils von einem Raspberry Pi repräsentiert werden. Mit diesem Aufbau werden verschiedene, realitätsnahe Szenarios simuliert.

Ein weiterer Aspekt der in Zukunft untersucht werden kann, sind die Vor- und Nachteile des Einsatzes von Virtualisierung im Kontext des V2X Application-Level Gateways. Ein mögliches Ziel zukünftiger Forschung sind auch die zum V2X Application-Level Gateway komplementären Paket-Filter Komponenten des V2X Security Gateways zur Absicherung der Kommunikation auf dem Internet- und Transport-Layer. Auch die sichere Speicherung kryptografischer Schlüssel, Zertifikate und Konfigurationen und deren Schutz vor Manipulation könnten zukünftig untersucht werden , da potentielle Angreifer physischen Zugriff auf das Fahrzeug haben können. Eine Bedrohung, auf welche das V2X Security Gateway bisher nicht eingeht, sind Denial-of-Service (DoS) Angriffe. Obwohl die meisten effektiven DoS-Gegenmaßnahmen Netzwerk-basiert sind [16], [11] und die Maßnahmen, welche ein einzelner Netzwerk-Knoten, in diesem Fall ein Fahrzeug, ergreifen kann, um sich vor DoS-Angriffen zu schützen beschränkt sind, wurde ein Mechanismus vorgeschlagen mit dem einzelne Netzwerk-Knoten zu ihrem Schutz vor DoS-Angriffen beitragen können. Das Grundprinzip hinter dem als *Hashcash* [2] bekannten Mechanismus besteht darin, dass jeder Interaktion mit einem Knoten, z.B. einem Connection-Request an den Knoten, Kosten zugewiesen werden. Konkret muss der Anfragende ein Token berechnen, damit der Knoten seine Anfrage bearbeitet (die Berechnung des Tokens basiert auf dem Finden partieller Hash-Kollisionen). Ein DoS-Angriff der den Knoten mit Anfragen überlastet wird somit sehr aufwändig, da jede Anfrage signifikante Ressourcen, in diesem Fall Rechenleistung, benötigt. Es könnte untersucht werden, ob Ansätze wie *Hashcash* sich für den Schutz vor DoS-Angriffen im Automobil-Bereich eignen.

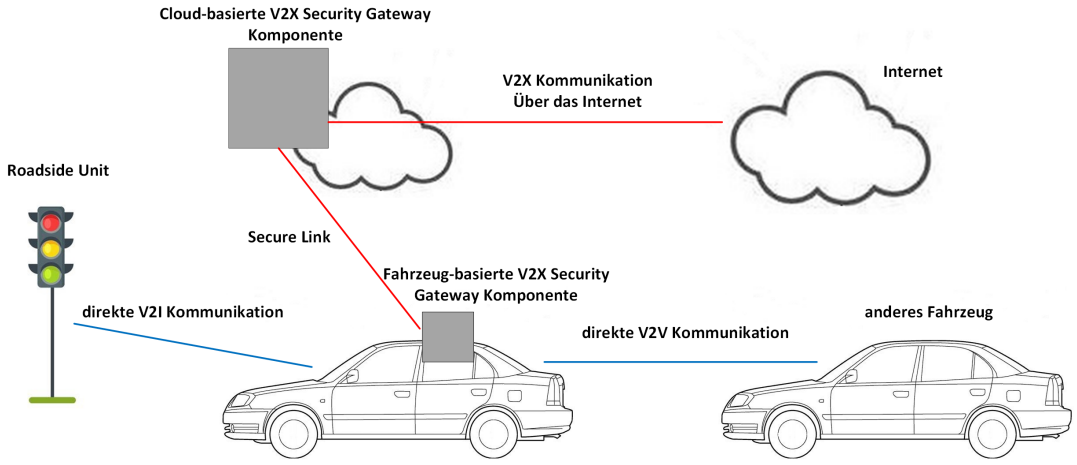

Abbildung 12. Cloud-basiertes V2X Application-Level Gateway

LITERATUR

[1] Apvrille, L., El Khayari, R., Henniger, O., Roudier, Y., Schweppe, H., Seudié, H., ... & Wolf, M. (2010, May). Secure automotive on-board electronics network architecture. In FISITA 2010 world automotive congress, Budapest, Hungary (Vol. 8).

[2] Back, A. (2002). Hashcash-a denial of service counter-measure.

[3] Bellovin, S. M., & Cheswick, W. R. (1994). Network firewalls. IEEE communications magazine, 32(9), 50-57.

[4] Bittl, S. (2017). Efficient Secure Communication in VANETs under the Presence of new Requirements Emerging from Advanced Attacks.

[5] Boban, M., Kousaridas, A., Manolakis, K., Eichinger, J., & Xu, W. (2017). Use cases, requirements, and design considerations for 5G V2X. arXiv preprint arXiv:1712.01754.

[6] Bouard, A., Schanda, J., Herrscher, D., & Eckert, C. (2012, November). Automotive proxy-based security architecture for ce device integration. In International Conference on Mobile Wireless Middleware, Operating Systems, and Applications (pp. 62-76). Springer, Berlin, Heidelberg.

[7] Brose, G. (2003). A gateway to web services security–Securing SOAP with proxies. In Web Services-ICWS-Europe 2003 (pp. 101-108). Springer, Berlin, Heidelberg.

[8] Bundesamt für Sicherheit in der Informationstechnik (BSI): Sichere Anbindung von lokalen Netzen an das Internet (ISi-LANA), BSI-Standards zur Internet-Sicherheit (ISi-S), Version 2.1 vom 26.08.2014

[9] Burnside, M., Clarke, D., Mills, T., Maywah, A., Devadas, S., & Rivest, R. (2002, March). Proxy-based security protocols in networked mobile devices. In Proceedings of the 2002 ACM symposium on Applied computing (pp. 265-272). ACM.

[10] Calandriello, G., Papadimitratos, P., Hubaux, J. P., & Lioy, A. (2007, September). Efficient and robust pseudonymous authentication in VANET. In Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks (pp. 19-28). ACM.

[11] Dietzel, C., Smaragdakis, G., Wichtlhuber, M., & Feldmann, A. (2018, December). Stellar: network attack mitigation using advanced blackholing. In Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies (pp. 152-164). ACM.

[12] Glass, M., Herrscher, D., Meier, H., & Schoo, P. (2010). "Seis"—security in embedded IP-based systems. ATZelektronik worldwide, 5(1), 36-40.

[13] Ghosh, M., Varghese, A., Kherani, A. A., & Gupta, A. (2009, April). Distributed misbehavior detection in VANETs. In 2009 IEEE Wireless Communications and Networking Conference (pp. 1-6). IEEE.

[14] Hao, Y., Cheng, Y., Zhou, C., & Song, W. (2011). A distributed key management framework with cooperative message authentication in VANETs. IEEE Journal on selected areas in communications, 29(3), 616-629.

[15] Idrees, M. S., Schweppe, H., Roudier, Y., Wolf, M., Scheuermann, D., & Henniger, O. (2011, March). Secure automotive on-board protocols: a case of over-the-air firmware updates. In International Workshop on Communication Technologies for Vehicles (pp. 224-238). Springer, Berlin, Heidelberg.

[16] Jonker, M., King, A., Krupp, J., Rossow, C., Sperotto, A., & Dainotti, A. (2017, November). Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. In Proceedings of the 2017 Internet Measurement Conference (pp. 100-113). ACM.

[17] Jourdan, G. V. (2007). Centralized web proxy services: Security and privacy considerations. IEEE Internet Computing, (6), 46-52.

[18] Luotonen, A., & Altis, K. (1994). World-wide web proxies. Computer Networks and ISDN systems, 27(2), 147-154.

[19] MacHardy, Z., Khan, A., Obana, K., & Iwashina, S. (2018). V2X access technologies: Regulation, research, and remaining challenges. IEEE Communications Surveys & Tutorials, 20(3), 1858-1877.

[20] Morabito, R., Petrolo, R., Loscri, V., & Mitton, N. (2018). LEGIoT: a Lightweight Edge Gateway for the Internet of Things. Future Generation Computer Systems, 81, 1-15.

[21] Nugur, A. (2017). Design and Development of an Internet-Of-Things (IoT) Gateway for Smart Building Applications (Doctoral dissertation, Virginia Tech).

[22] Pesé, M. D., Schmidt, K., & Zweck, H. (2017). Hardware/Software Co-Design of an Automotive Embedded Firewall (No. 2017-01-1659). SAE Technical Paper.

[23] Pretschner, A., Broy, M., Kruger, I. H., & Stauner, T. (2007, May). Software engineering for automotive systems: A roadmap. In Future of Software Engineering (FOSE'07) (pp. 55-71). IEEE.

[24] Ruj, S., Cavenaghi, M. A., Huang, Z., Nayak, A., & Stojmenovic, I. (2011, September). On data-centric misbehavior detection in VANETs. In 2011 IEEE Vehicular Technology Conference (VTC Fall) (pp. 1-5). IEEE.

[25] Schmidt, D. C. (1996). A family of design patterns for applications-level gateways. TAPOS, 2(1), 15-30.

[26] Schunter, M., et al. (2017, November). Vehicle to Cloud - Research Challenges for Intelligent Vehicles. 15th Escar Europe Conference, Berlin.

[27] Scott, D., & Sharp, R. (2002, May). Abstracting application-level web security. In Proceedings of the 11th international conference on World Wide Web (pp. 396-407). ACM.

[28] Szancer, S. (2018, October). Architektur eines V2X Automotive Security Gateways (Report, Hamburg University of Applied Sciences).

[29] Wei, D., Darie, F., & Shen, L. (2013, February). Application layer security proxy for smart Grid substation automation systems. In Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES (pp. 1-6). IEEE.

[30] Weimerskirch, A. (2011, October). V2X security & privacy: the current state and its future. In ITS World Congress, Orlando, FL.

[31] Wijnants, M., & Lamotte, W. (2007, June). The NIProxy: a Flexible Proxy Server Supporting Client Bandwidth Management and Multimedia Service Provision. In World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a (pp. 1-9). IEEE.

[32] Yang, X., Liu, L., Vaidya, N. H., & Zhao, F. (2004, August). A vehicle-to-vehicle communication protocol for cooperative collision warning. In Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on (pp. 114-123). IEEE.