



CHALMERS



IT-säkerhet i uppkopplade fordon

En analys av säkerhetsbilden på AGA-plattformen

Examensarbete på högskoleingenjörsprogrammet i datateknik

ALEXANDER BONDEFALK

DAVID SVENSSON

EXAMENSARBETE

IT-säkerhet i uppkopplade fordon

En analys av säkerhetsbilden på AGA-plattformen

ALEXANDER BONDEFALK
DAVID SVENSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Göteborg 2015

IT-säkerhet i uppkopplade fordon

En analys av säkerhetsbilden på AGA-plattformen

ALEXANDER BONDEFALK

DAVID SVENSSON

©A.BONDEFALK, D.SVENSSON, 2015

Examinator: Lars Svensson

Institutionen för Data- och Informationsteknik

Chalmers Tekniska Högskola

412 96 Göteborg

Telefon: 031-772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Omslag: Visar AGAs koncept, uppkopplad android i fordon.

Institutionen för Data- och Informationsteknik

Göteborg 2015

Förord

Denna rapport är resultatet av ett examensarbete utfört av Alexander Bondefalk och David Svensson på Dataingenjörsprogrammet, Högskoleteknologiesektionen. Examinatorn för examensarbetet var Lars Svensson. Examensarbetet innefattar 15 högskolepoäng och är beställt av Combitech AB med mål att analysera IT-säkerheten i den öppna fordons mjukvaruplattformen AGA. Då arbetet utförs i utbildningssyfte så har rapporten blivit prioriterad på samma nivå som projektresultatet.

Ett särskilt tack utfärdas till handledaren på Combitech, Fredrik Holgersson och hans medarbetare Emilie Barse samt Christopher Pavlic och vår handledare Tomas Olovsson för all hjälp under arbetets gång. Vi vill även rikta ett tack mot Combitech och Volvo med all involverad personal för att ha gjort detta arbete möjligt.

IT-säkerhet i uppkopplade fordon

En analys av säkerhetsbilden på AGA-plattformen

ALEXANDER BONDEFALK

DAVID SVENSSON

Institutionen för Data- och Informationsteknik, Chalmers Tekniska Högskola

Examensarbete

Sammanfattning

Detta examensarbete är utfört på begäran av Combitech AB och behandlar säkerhetsbrister i plattformen Automotive Grade Android, AGA, hur dessa brister kan utnyttjas och sedermera lösas. Vidare beskrivs två större säkerhetslösningar som skulle förbättra säkerheten och pålitligheten i AGA.

Studien presenterar en omfattande lista över nuvarande och framtida gränssnitt mot AGA och studerar de vanligaste felen och bristerna i dessa. Mer specifikt så visar studien på flera problem, arkitekturella och implementationsmässiga, där kryptering och virtualisering föreslås som lösningar på dessa. Tillsammans med information från bland annat intervjuer med relaterade företag diskuteras dessa i mer detalj.

I fallet av virtualisering så studeras cost-reward och trenden med hypervisor som säkerhetslösning. I fallet av kryptering så diskuteras teoretiskt problem i den nuvarande arkitekturen och en lösning på detta presenteras.

Abstract

This thesis was commissioned by Combitech and explores security flaws in the platform Automotive Grade Android, AGA, and how these can be exploited and later solved. The thesis will present the flaws that have been found, the security mechanisms that have been identified as well as those that are recommended to be implemented.

Furthermore, the thesis describes a comprehensive list of current and future interfaces of vehicle communication and common security flaws in them. The study shows several problems, architectural and implementational, and suggests encryption and virtualization as solutions for these, which are discussed in detail with information from interviews with related companies.

In the case of virtualization a comparison of cost-reward versus hardware solutions, if the cost of a dedicated hardware unit is worth the added security, as well as the trend with hypervisors as a security solution. Encryption is theoretically discussed concerning the problems in the current architecture and presents a solution to those.

1 Innehåll

1	Inledning	2
1.1	Bakgrund	2
1.1.1	Säkerhet	2
1.1.2	AGA - Automotive Grade Android	2
1.1.3	Sårbarhet i fordon	3
1.1.4	Android i fordon	3
1.1.5	Syftet bakom AGA	3
1.2	Syfte	4
1.3	Avgränsningar	4
1.4	Problemställning	4
1.5	Rapportens struktur	4
2	Teknisk bakgrund	6
2.1	Kommunikation och operativsystem i fordon	6
2.1.1	CAN	6
2.1.2	Gateway	6
2.1.3	AUTOSAR	6
	Virtualisering	6
2.2	6
2.2.1	Xen hypervisor	6
2.2.2	TrustZone	6
	Kryptering och Datasäkerhet	7
2.3	7
2.3.1	Replayattack	7
2.3.2	Asymmetrisk kryptering	7
2.3.3	Symmetrisk kryptering	7

2.3.4	Strömkryptering.....	7
2.3.5	IDS - Intrusion Detection System.....	8
 Arkitekturen i AGA	
	9
3	9
3.1	Automotive Service	10
3.2	Policy Manager.....	10
3.3	Noder och Tuber.....	10
3.4	Proxy.....	11
3.5	Applikationen.....	11
3.6	Vehicle Integration Layer (VIL).....	13
3.7	System Data Protocol (SDP).....	13
3.7.1	Signaler.....	13
3.7.2	Kommunikation mellan SDP-noder	14
3.8	Intern Kommunikation	14
3.9	Topologi.....	15
 Gränssnitt för kommunikation	
	17
4	17
	Gränssnitt mot fordonet.....	17
4.1	17
	Bluetooth.....	17
4.1.1	17
4.1.2	4G.....	17
4.1.3	USB.....	18
4.1.4	AUX.....	18
4.1.5	CD-spelare.....	19
4.1.6	WiFi.....	19
4.1.7	HMI.....	19
4.2	Protokoll.....	19
4.2.1	TCP/IP	19

4.2.2	SDP svagheter	20
4.2.3	Protokoll för användarapplikationer	20
5	Metod.....	22
5.1	Metoder för datainsamling.....	22
5.1.1	Instudering	22
5.1.2	Personliga intervjuer	22
5.1.3	Fallstudie	22
5.2	Design av experiment.....	23
5.2.1	Design av intervjuer	23
5.2.2	Design av fallstudier.....	23
5.2.3	Basfall.....	23
5.3	Verktyg.....	23
5.3.1	Kodhanteringsverktyg.....	23
5.3.2	Säkerhetsverktyg.....	24
5.3.3	Övriga.....	24
6	Genomförande.....	25
6.1	Uppstart.....	25
6.2	Inledande intervjuer	25
6.3	Fallstudie med arkitekturella svagheter	25
6.4	Uppföljande intervjuer.....	25
6.5	Fallstudie med nätverkssvagheter	25
6.6	Fallstudie om reboot.....	26
6.7	Ta fram förslag.....	26
6.8	Avslutande presentationer	26
7	Resultat	27
7.1	Identifierade brister och hur de kan utnyttjas	27
7.1.1	Fordonet är ej skyddad mot ett komprometterat AGA.....	27
7.1.2	Icke infångat undantag.....	28
7.1.3	Informationssårbarhet mellan nätverk	28
7.1.4	SDP DoS.....	28
7.1.5	SDP Replay	30

7.2	Existerande säkerhetsmekanismer och hur de kan förbättras.....	30
7.2.1	Policy manager.....	30
7.2.2	Robusthet i AGA:s arkitektur.....	30
7.3	Lämpliga säkerhetsmekanismer och hur dessa skall implementeras.....	30
7.3.1	Gateway.....	30
7.3.2	Kryptering & Signering.....	32
7.3.3	Nyckelhantering.....	33
7.3.4	Autentisering.....	33
7.3.5	Signalhantering.....	33
7.3.6	IDS.....	33
8	Slutsats.....	34
8.1	Förslag på implementering av virtuell gateway.....	34
8.2	Förslag på implementation av kryptering.....	36
8.2.1	Önskade funktioner.....	36
8.2.2	Förlsag på topologier.....	37
8.2.3	CAN och kryptering.....	39
9	Diskussion.....	42

Beteckningar

ADB - Android Debug Bridge
AES - Advanced Encryption Standard
AGA - Automotive Grade Android
AOSP - Android Open Source Project
ARM - Advanced RISC Machine
AUX-port - Auxiliary-port
Binder IPC - Binder Inter Process Communication
CAN - Controller Area Network
CPU - Central Processing Unit
DoS - Denial-of-Service
ECU - Electronic Control Unit
FTP - File Transfer Protocol
HMI - Human Machine Interface
HTTP - HyperText Transfer Protocol
HVM - Hardware-assisted virtualization
IDE - Integrated Developer Environment
IDS - Intrusion Detection System
IP - Internet protocol
IPC - Inter Process Communication
JVM - Java Virtual Machine
NIO - Java Non Blocking Input/Output
NIST - National Institute of Standards and Technology
NMAP - Network Mapper
NVD - National Vulnerability Database
OEM - Original Equipment Manufacturer
OS - Operativsystem
OSEK - Open Systems and their Interfaces for the Electronics in Motor Vehicles
PV-ParaVirtualisering
RTOS - Real-Time operating system
SCS - Seamless Communication System
SDP - System Data Protocol
TCP - Transmission control protocol
TPM - Trusted Plattform Module
USB - Universal serial bus
VIL - Vehicle integration layer
VICTA - Vehicle ICT Arena
QSEE - Qualcomm Secure Execution Environment

1 Inledning

I denna rapport genomförs en säkerhetsanalys av mjukvaruplattformen AGA, Automotive Grade Android. Detta följs av ett förslag på en lösning där de sårbarheter som analysen tar upp diskuteras och lämpliga säkerhetsmekanismer identifieras.

1.1 Bakgrund

Fordonsutveckling är idag inte standardiserat. En lösning kan fungera för en specifik tillverkares bilserier eller ibland till och med bara för ett enskilt fordon. Tanken med Automotive Grade Android, AGA, är att tillåta en välkänd plattform, Android, med tillgång till en stor mängd utvecklare, att ta del av de system som finns tillgängliga i fordonen, så som temperaturmätare, hastighetsmätare och annan data. För en förare ska det vara möjligt att ge sitt fordon personliga inställningar som är av dennes intresse. Genom att kommunicera och läsa av sin omgivning ska fordonet ta in information och använda den för att förbättra förarupplevelsen på ett säkert sätt. För att detta ska fungera krävs en stabil mjukvaruplattform som ska göra det möjligt för utvecklare att hämta och skicka information från fordonets CAN-buss utan att äventyra säkerheten hos fordonet. Eftersom dessa system även kan vara värdefulla för en angripare, äventyrar det säkerheten i fordonet och ställer därför höga krav på mjukvaruplattformen i fordonet.

1.1.1 Säkerhet

Begreppet säkerhet har två aspekter, safety och security. Safety, som betyder att fordonet skall vara säkert vid olyckor och inte utgöra en fara för föraren eller omgivningen, är en viktig parameter inom fordonsindustrin. Samtidigt som fordonens teknik utvecklas och expanderar ökar även dess attackytor vilket gör att security, vilket innebär säkerhet vid intrång eller informationsförlust, också blir viktig. Attacker behöver i dagens läge inte utföras på plats utan kan istället ske mot ett uppkopplat fordon, vilket även gör det möjligt för storskaliga attacker, till exempel då angriparen använder sig av svagheter i ett fordons bluetooth för att få obehörig tillgång till fordonets funktioner [18]. Detta kan leda till att information läcks eller att kritiska funktioner inte fungerar. Därför är det viktigt att motverka intrång i mjukvaruplattformen vilket annars kan äventyra säkerheten i fordonet.

1.1.2 AGA - Automotive Grade Android

Tillsammans med bl.a. Swedspot fick Combitech i uppdrag av Vehicle ICT Arena att skapa en plattform som möjliggör kommunikation mellan Android och inbyggda system i fordon. I augusti 2014 släppte de mjukvaruplattformen AGA[1]. Automotive Grade Android är en open source mjukvaruplattform som använder välkända verktyg för att tillåta en utvecklare att integrera Androidapplikationer med ett fordons infotainmentsystem. Ett exempel skulle kunna vara en parkeringsapplikation som tillåter fordonet att kommunicera med parkeringsplatserna för att leta efter den närmsta lediga parkeringsplatsen för fordonet och betala för den.

1.1.3 Sårbarhet i fordon

Ett flertal rapporter såsom ”*Adventures in automotive networks and controll units*” av Chris Valsek och Charlie Miller [2] samt ”*Playing with Car Firmware or How to Brick your Car*” av Paul Such [3] har bevisat stora säkerhetsbrister i CAN-systemet. Trots att modernare fordon har fler än ett CAN-nätverk som är uppdelade med gateways är det snarare på grund av den ökade komplexiteten i fordonens funktioner än från ett säkerhetsperspektiv som dessa ändringar har genomförts[4]. Majoriteten av de gränssnitt som använts för att få tillgång till fordonet har sina rötter i infotainmentsystemet som skulle kunna flyttas från att ligga separat på CAN-bussen och istället samlas i en eller flera Androidstyrda enheter. Till exempel i ”*A Survey of Remote Automotive Attack Surfaces*” [4], också av C. Miller och C. Valsek, beskrivs Bluetooth, radion, WiFi och särskilt Internet och applikationer som intrångsvägar in i fordonet. Flera av dessa är också, som C. Miller och C. Valsek tar upp, onödiga att koppla till ett system så viktigt som CAN. Kopplingarna har gjorts främst för att CAN-systemet redan existerade. Som exempel så bör radion knappast ha viktiga meddelanden att skicka till motorn.

1.1.4 Android i fordon

I en vitbok utgiven av Wind River ”*Common Android security vulnerabilities in an automotive environment*” [5] diskuteras Android som en integrerad plattform för fordon och att omodifierad Android inte är lämplig till att kopplas direkt till CAN-bussen på grund av säkerhetskäl. Samtidigt så listar amerikanska NVD, National Vulnerability Database, 1,757 sårbarheter i Android de senaste tre åren [6]. Wind River säger i sin vitbok att kombinationen av sårbarheter i Android tillsammans med att utvecklare sällan följer alla säkerhetsriktlinjer kommer resultera i problem i en fordonsmiljö. Men däremot menar Wind River att Android kan vara en bra startpunkt för vidare utveckling. Med tillagda säkerhetsmekanismer i båda hårdvara och mjukvara, så som skyddsmekanismer för användarapplikationer och säkerhetsinriktad kodgranskning, kan en stabil och säker miljö skapas.

1.1.5 Syftet bakom AGA

AGA är skapat med ”*specifika regler som säkerställer att applikationerna kan användas på ett säkert sätt i fordon*” [1] och lämnar hårdvaran samt fordonskommunikationen till OEM:en för att utveckla genom VIL (se 3.6). Med en utomstående kontrollfunktion till exempel en filtrerande gateway som på ett säkert sätt kan hantera den data som flödar mellan nätverken, så löser detta både problemen med sårbarheterna i fordonet (se 1.1.3) och att en omodifierad Android inte är lämplig att kopplas direkt emot CAN (se 1.1.4) då infotainmentsystemen blir separerade från ECU:er i CAN och inlagda i Androidmiljön i AGA som ska ha säkerhetsmekanismer inplanerade för att lösa problem som är typiska med Android.

1.2 Syfte

Syftet med detta examensarbete är att för företaget Combitech genomföra en säkerhetsanalys mot mjukvaruplattformen AGA med målsättningen att undersöka vilka sårbarheter samt buggar som påverkar säkerheten. I denna rapport analyseras flera gränssnitt som kan fungera som attacktytor för systemet, främst öppna portar på enheten och ett förslag för att lösa flera av de säkerhetsbrister som idag existerar i AGA presenteras. Detta kommer tillåta en utvecklare att lösa många av de säkerhetsbrister och svagheter som annars skulle ha existerat.

1.3 Avgränsningar

Då säkerhet kan utvärderas och undersökas i olika skalor, hela vägen till fullskaliga säkerhetscertifieringar har vissa avgränsningar gjorts. Denna rapport baseras på de implementationer som vi har haft tillgång till, vilket innebar en reducerad förmåga att utvärdera den slutliga produkten eftersom arbetet utgick från en prototyp av AGA. Detta innebär att även om samtliga gränssnitt som är vanligt förekommande i fordon beskrivs i rapporten är inte alla behandlade med stor noggrannhet då vi inte har tillgång till ett fullt utrustat fordon. Då vi inte har haft tillgång till ett AGA-system installerat i ett fordon, så har vi analyserat ett AGA-system som har varit installerat i en surfplatta med fokus på AGA som en attackyta.

1.4 Problemställning

AGA ska kunna kommunicera säkert med fordonet och ta del av dess resurser utan att vara sårbart för angrepp, varken mot gränssnittet som ligger mot föraren, så som applikationer och internettrafik, eller det gränssnitt som ligger mot fordonet, främst CAN-bussen. AGA ska även kunna reagera mot distraherade förare genom att ändra nivån av stimulation som infotainmentsystemet tillhandahåller, vilket bland annat gör att ljud sänks eller stängs av beroende på situationen. Detta ska inte kunna gå att avbryta.

Följande problem behandlas i detta arbete:

- 1. Vilka gränssnitt och attacktytor existerar i AGA och hur kan de utnyttjas?*
- 2. Vilka existerande säkerhetsmekanismer finns, vilka brister har de, och hur kan de förbättras?*
- 3. Vilka säkerhetsmekanismer skulle vara lämpliga att implementera i AGA och hur?*

1.5 Rapportens struktur

I nästa kapitel görs en kort introduktion till de problem som infotainment, samt vanlig Android har samt några av de tekniska termer som behövs för förståelse av rapporten. I kapitel 3 beskrivs vidare AGA:s specifika arkitektur, hur AGA fungerar och hur det används.

Kapitel 4 beskriver de gränssnitt som AGA har samt de som ofta finns i fordon. Alla kapitel fram till denna menar att beskriva bakgrunden av arbetet. Kapitel 5 förklarar de datainsamlingsmetoder och den experimentdesign som har implementerats under arbetet. Kapitel 6 går stegvis igenom arbetet och förklarar i varje steg hur arbetet har utförts. Kapitel 7 presenterar arbetets resultat, här besvaras de frågeställningar som arbetet ämnade lösa. Kapitel 8 beskriver möjliga lösningar för de säkerhetsbrister som har hittats. Kapitel 9 utgörs av diskussion kring hur arbetet har utförts, de resultat som har funnits och de lösningar som har presenterats. Diskussionen går även in på hur arbetet har utförts och det som kunde ha gjorts annorlunda.

2 Teknisk bakgrund

2.1 Kommunikation och operativsystem i fordon

2.1.1 CAN

CAN (Controller Area Network) är ett seriellt bussnätverk för bilar och dess funktion är att kunna koppla ihop flera noder på en och samma buss. De data som kommer från noderna skickas som en samling av bit-signaler. CAN utvecklades år 1983 och kunde då skicka data med en hastighet på 125kbit/s, i dagens läge kan det skickas i 500kbit/s. För att öka pålitligheten av ett meddelande använder sig CAN av en identifierare för att sätta prioritering på meddelanden. Genom att använda sig av en mekanism för arbitrerings kan CAN hantera konflikter och på så sätt garantera att meddelanden inte kolliderar med varandra så att information går förlorad. Detta har gett att fordonsindustrin haft stor användning av protokollet då ett fordon innehåller tidskritiska system som t.ex. en bromssignal. [48]

2.1.2 Gateway

En gateway är en kommunikationsenhet som kopplar ihop två nätverk. All information som skickas mellan nätverken går genom denna gateway. En gateway består av mjukvara eller hårdvara och kan i vissa fall också fungera som en brandvägg som kan filtrera bort skadlig information som skickas mellan nätverken. [49]

2.1.3 AUTOSAR

AUTOSAR är ett standardiserat reelltidsoperativsystem som utvecklats för att köras i fordon. Den möjliggör integration med fordonets system. Det är baserat på en standard kallad OSEK som skapat förteckningar för ett inbyggt OS [42].

2.2 Virtualisering

2.2.1 Xen hypervisor

Xen projektet är en Open-Source bare metal, eller nivå 1, hypervisor. Detta är i dagsläget ett samarbetsprojekt som drivs av the Linux foundation [37]. Xen styr över minneshantering och CPU. Den stödjer två olika virtualiserings sätt: hårdvaru-assisterad virtualisering(HVM) och hårdvaru-assisterad virtualisering med Paravirtualisering(PV). HVM tillåter omodifierade gäster vilket betyder att inga ändringar behöver göras på ett operativsystem för att det ska fungera. Paravirtualisering tillåter körning av modifierade operativsystem detta för att öka prestandan för dess värd. Dessa kan användas samtidigt på samma hypervisor [38].

2.2.2 TrustZone

TrustZone introducerades i ARM arkitektur version 6. Det är en teknologi utvecklad för att göra ARM-processorer "trovärdiga". Tanken bakom TrustZone är att skapa ett särskilt CPU

läge “secure mode”. Detta delar upp till två världar, “normal värld” och “säker värld”. Den säkra världen kan utföra både minnesaccesser klassade som säkra och även icke-säkra accesser. Den normala världen kan endast utföra icke-säkra accesser. Målet med detta är att isolera säker mjukvara i en simpel miljö som inte är direkt sårbar mot mjukvarusårbarheter i enhetens huvudoperativsystem [40]. Att ha en TrustZone som kör vid sidan av enhetens operativsystem är fördelaktigt då utvecklaren kan klassa hårdvaruresurser som säkra eller normala. Vid implementationen av TrustZone i QSEE, Qualcomm Secure Execution Environment, uppstod en sårbarhet som gjorde det möjligt för en angripare att få QSEE att skriva data till det säkra minnet [41].

2.3 Kryptering och Datasäkerhet

2.3.1 Replayattack

En replayattack innebär ett meddelande återsänds av en angripare utan att denne har tillgång till krypteringsnyckeln eller dess innehåll. Attacken fungerar om meddelanden alltid är giltiga. Ett skydd mot denna typ av attack måste på något sätt göra ett meddelande ogiltigt efter att det är mottaget eller att det expirerar efter en viss tid. Ett sätt att göra detta är att använda en räknare över vilket meddelandenummer som utbyts och efter att meddelandet skickats väntar sändaren på ett svar med nästa nummer i följd.

2.3.2 Asymmetrisk kryptering

Asymmetrisk kryptering innebär att ett meddelande krypteras och dekrypteras med två olika nycklar. Detta kallas Public-key cryptography och gör att en person, 'Alice', kan skicka sin publika nyckel till en annan person, 'Bob', och att 'Bob' sedan kan kryptera och skicka meddelanden till 'Alice' som endast hon kan dekryptera med sin privata nyckel.

2.3.3 Symmetrisk kryptering

Symmetrisk kryptering innebär att ett meddelande endast kan sändas och tas emot med en gemensamt överenskommen nyckel. Detta innebär att varje session har sin egen nyckel som måste innehas för att kunna tyda meddelandena. Symmetriska nycklar används främst då asymmetriska nycklar är för långsamma för ändamålet, till exempel i ett fordon med begränsad beräkningskraft.

2.3.4 Strömkryptering

Strömkryptering innebär att två världar som kommunicerar alltid är medvetna om vilket nummer i meddelandeföljden som ska skickas eller tas emot. Detta gör att båda världarna uppdaterar sin symmetriska nyckel utan kommunikation med den andre vilket gör det svårt för en angripare att bryta kommunikationen. Ett förenklat exempel lyder: en kryptering av 123 blir 321, med ett sekvensnummer 000, skulle göra att 123 blir 321000 första gången och

321001 andra gången, det skulle då inte gå att utföra en replayattack då sekvensnumret skulle vara felaktigt. [50]

2.3.5 IDS - Intrusion Detection System

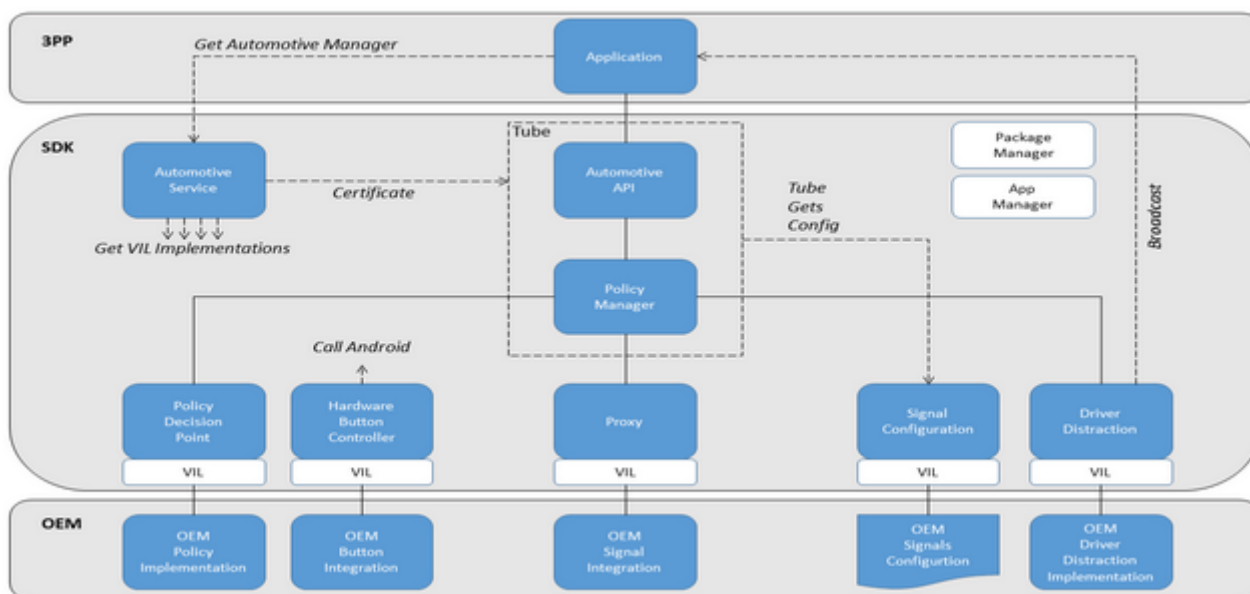
Ett Intrusion Detection System är ett mjukvaruprogram som undersöker händelser på ett datorsystem eller nätverk och analyserar dem efter tecken på intrång i systemet. Ett IDS kan antingen placeras på en värd-enhet, där den fokuserar på att undersöka inkommande och utgående trafik, eller på ett nätverk, där den fokuserar på att analysera trafik som sker med användare inifrån nätverket. Ett IDS har endast som uppgift att samla in information ifrån ett intrång, inte motverka det. [35]

3 Arkitekturen i AGA

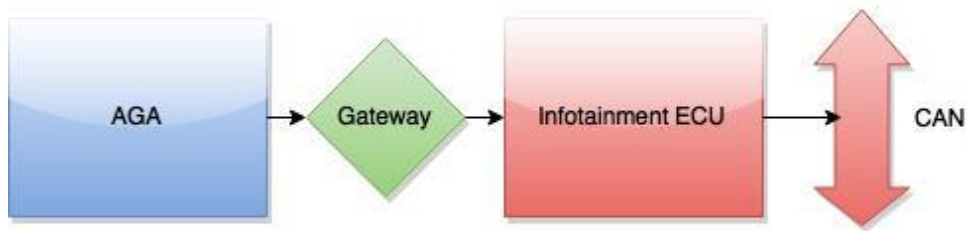
Följande komponenter ingår i AGA arkitekturen:

1. **Automotive Service:** Huvudkomponenten som har hand om all meddelandehantering i Android. Anslutande applikationer kan kalla denna systemstjänst för att få ta del av meddelanden i AGA-nätverket.
2. **Policy Manager:** Kod som vaktar applikationer och bestämmer vilka signaler de får skicka.
3. **Noder & Tuber:** Mjukvarurepresentationer av applikationer och hårdvara i AGA.
4. **Proxy:** Den knutpunkt i AGA som hämtar information från fordonet, förutom i vissa specialfall som hårdvaruknappar.
5. **Applikationer:** Tredjepartsprogram som använder AGA.
6. **VIL:** Gränssnittet mellan fordonet och AGA som översätter SDP-kommunikation till någonting som bilen förstår och vice versa.
7. **SDP:** SDP är det meddelandeprotokollet som AGA använder för att kommunicera mellan noder. Detta är ett centralt koncept i AGA.
8. **Intern Kommunikation:** Beskriver de olika sätt AGA kommunicerar, såsom Inter-process kommunikation.
9. **Topologi:** Beskriver AGA's topologi.

Figur 1 presenterar en översikt över de olika komponenter som finns i AGA och hur dessa är relaterade till varandra och figur 2 visar en förenklad översikt av hur AGA är sammankopplad med CAN-bussen på ett fordon.



Figur 1: Översikt av AGAs Arkitektur och dess komponenter[1]



Figur 2: Förenklad vy av AGAs koppling till CAN-bussen

3.1 Automotive Service

I AGA som är en modifierad version av Android är Automotive Service den största skillnaden mot ett omodifierat Android. Automotive Service är en systemtjänst som är igång så länge AGA är igång. Automotive Service är ansvarig för att skapa Automotive Managers med vilka applikationer kan kommunicera med sina tuber och passa dessa vidare till de applikationer som ansöker om dem. Automotive Manager har även samma ansvar för VIL-instanser (se 3.6).

Automotive Service öppnar tre portar på Androidenheten:

- 8251, som hanterar interna SDP-meddelanden och även tar emot SDP-meddelanden från simulatorn.
- 9898, som hanterar Driver Distraction, vilket ställs in efter hur mycket föraren måste koncentrera sig på vägen. Dessa skickas som Android Broadcasts.
- **9899, som används för hårdvaruknappar.**

3.2 Policy Manager

Då en OEM väljer att använda AGA blir den även ansvarig för att ge ut certifikat till dess applikationer. Certifikaten används för att bestämma vilka rättigheter dessa applikationer har, såsom att kopplas till AGA-nätverket och att skicka eller ta emot olika signaler. Policy Manager implementeras av OEM för att sätta regler för tuber.

3.3 Noder och Tuber

En nod är en AGA-representation av ett objekt; detta kan vara en applikation eller representation av en hårdvarufunktion som en högtalare (grå och gröna cirklar i figur 3). Noder kan skicka och ta emot trafik och genom tillgång till en nod så kan även en applikation skicka och mottaga trafik.

En tub är en instans av en nod vaktad av en Policy Manager, skapad för en applikation. Tuben hanterar de signaler som applikationen har tillåtelse att skicka och ta emot. En vanlig nod är ej vaktad på samma sätt; de kan alltså skicka och ta emot alla signaler. En sådan nod kan vara en representation av någonting i bilen, såsom ett ljudreglage vilket skickar signaler eller en högtalare, som tar emot signaler.

3.4 Proxy

Proxyns uppgift är att kommunicera med fordonet. Trafik som går genom AGA in eller ut från fordonet passerar genom proxyn, vilken kommunicerar med bilen via VIL-lagret.

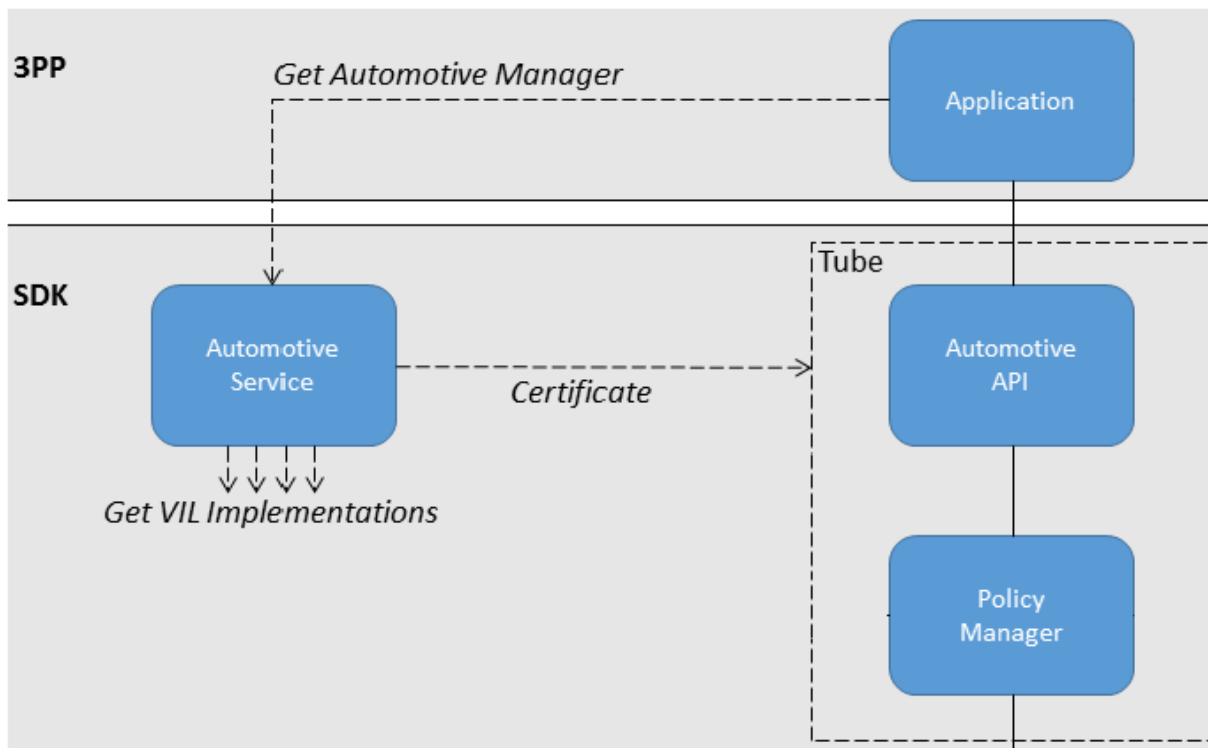
3.5 Applikationen

En AGA-applikation, kompilerad med AGAs SDK, kan genom att skapa ett Automotive Manager-objekt och sätta detta som:

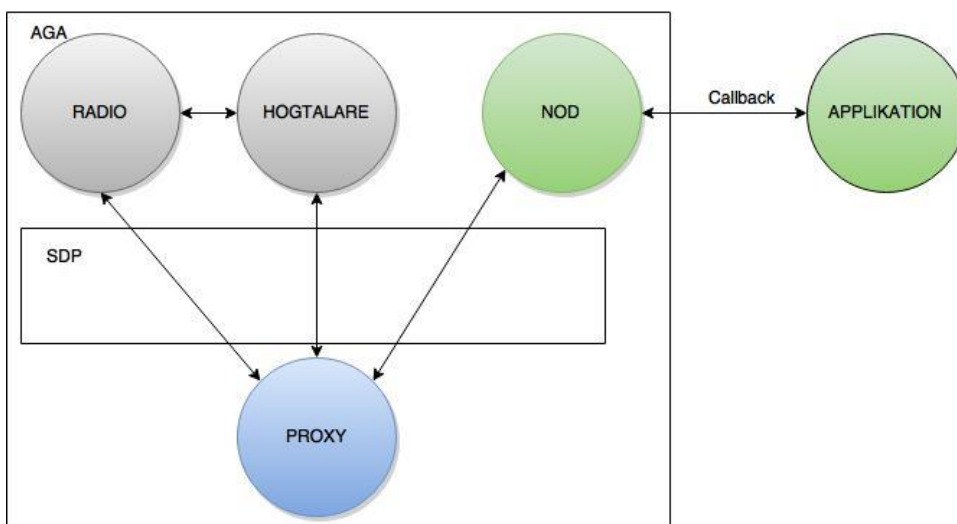
`getApplicationContext().getSystemService(Context.AUTOMOTIVE_SERVICE)` få tillgång till AGA-nätverket. I Figur 3 representeras denna fråga av pilen mellan Applikationen och Automotive Services. Automotive Service kommer förutom att skapa en Automotive Manager även skapa en tub som representerar applikationen i AGA-nätverket. Figur 3 visar hur tuben är ansluten till applikationen: Då information skickas som applikationen vill ha så tas denna emot av applikationens tub och hanteras sedan av applikationens Automotive Manager. Figur 4 visar ett exempel på hur ett AGA-nätverk ser ut och vad AGA-nätverket består av.

Att skapa en applikation kan exemplifieras med följande scenaria (och figur 3): Om en hastighetsmätare, applikationen, laddas ner till AGA så kommer den först att begära en Automotive Manager genom att kalla på Automotive Service. Applikationen får även en tub som den kan använda med hjälp av sin Automotive Manager. Sedan registrerar applikationen de signaler den önskar mottaga. I detta fall så vill den ha fordonets hastighet. För att hantera den information som nu ges så behövs en lyssnare i applikationen som kan ta informationen och låta applikationen visa upp det för användaren. Om hastighetsmätaren sedan märker att föraren kör för fort och vill uppmärksamma föraren om detta så kan den använda sin Automotive Manager för att skicka ett meddelande till högtalaren som informerar föraren om

detta.



Figur 3: Applikationens vy av AGA [1]



Figur 4: Exempel på ett AGA-nätverk: Intern kommunikation med SDP. Här finns två fysiska enheter (grå), en radio och en högtalare som kan kommunicera med varandra samt en applikations tub (grön) och extern kommunikation via lyssnare och funktioner. Då ett meddelande sänds mellan noder som ej har direktkoppling eller är hämtat från fordonet så skickas detta via proxyn.

3.6 Vehicle Integration Layer (VIL)

VIL står för Vehicle Integration Layer och är ett gränssnitt för att översätta information mellan AGA och fordonet. En OEM implementerar varje VIL för att fungera med deras fordon, så ett fullt fungerande AGA skapas.

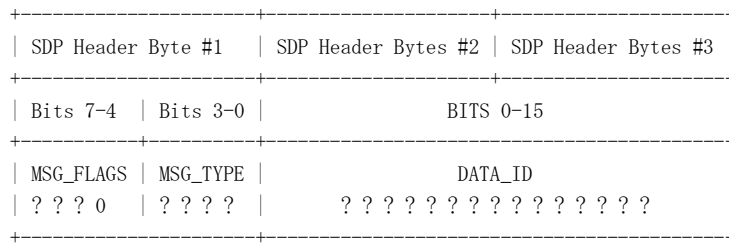
3.7 System Data Protocol (SDP)

AGA-nätverket är uppbyggt på av SDP noder som kommunicerar sinsemellan, internt eller genom andra medier som CAN eller TCP/IP. Eftersom SDP skapades för att efterlikna CAN är det enkelt att använda SDP över CAN-bussen och göra integrationen med fordon enklare. En nod i AGA kan producera och prenumerera på signaler, alltså lyssna efter och sända information. Endast en nod kan vara producent till en signal. Det är dock möjligt att skapa en nod som inte har en policy manager om man skapar en nod utan att kalla de funktioner som skapar en säker nod. Till exempel om en angripare skapar en applikation som använder källkoden för AGA, som är open-source, och inte bara det SDK som är för utvecklare för att skapa en nod eller om det är en systemnod som skapas av AGA vid uppstart.[8]

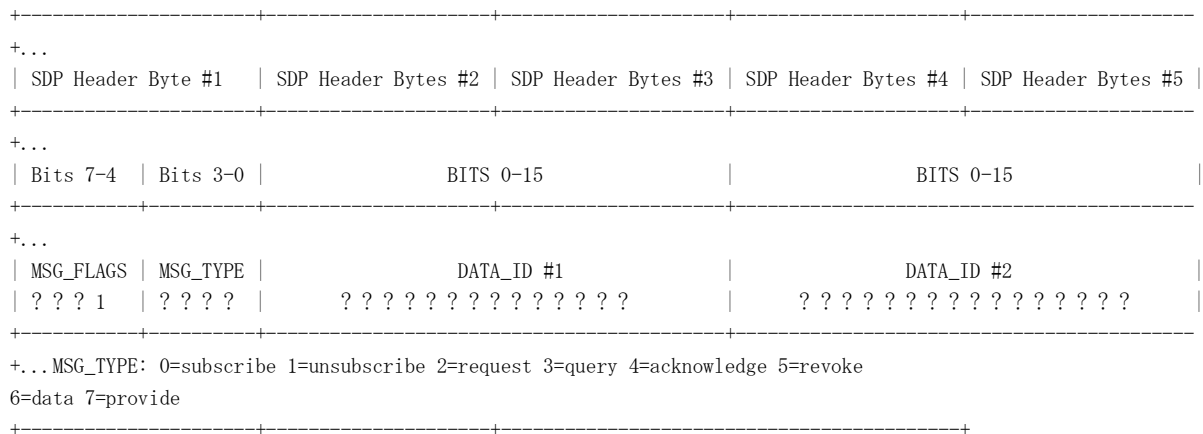
3.7.1 Signaler

Varje signal har ett 16-bitars identifikationsnummer kallat dataID som är unikt för varje enskild signal. DataID:t tillåter noder att känna igen signalen, och om noden ska läsa den, och vart signalen ska skickas. Till SDP så finns det även en wrapper, SCS som står för seamless communication system, som förenklar hanteringen av olika datatyper [8]. Figur 5 visar uppbyggnaden av SDP-meddelanden.

SDP Header (short header): 1 header byte followed by one 16-bit data id



SDP Header (long header): 1 header byte followed by many 16-bit data ids



Figur 5: Uppbyggnaden av SDP-meddelanden [1]

3.7.2 Kommunikation mellan SDP-noder

För att det ska vara möjligt att skicka data mellan SDP-noder som ligger i olika nätverk måste en AGA-gateway skapas för att koppla ihop kommunikationen. Gateway:en skickar data mellan två SDP-noder med hjälp av TCP/IP eller CAN (andra media kan konfigureras men är ej tillgängliga i dagsläget).[8]

3.8 Intern Kommunikation

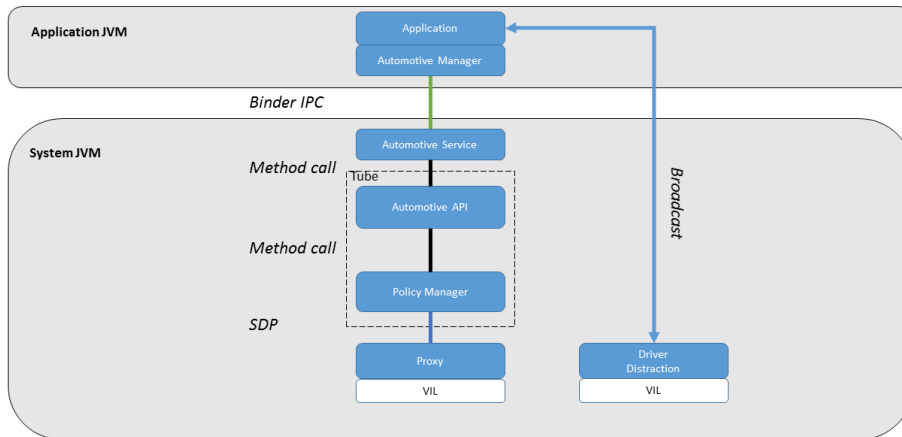
Det finns olika typer av intern kommunikation i AGA: Figur 6 ger ett detaljerat diagram över möjlig kommunikation inom AGA.

Binder IPC - Inter Process Communication

Binder IPC möjliggör kommunikation mellan olika Java Virtual Machines (JVM) genom ett delat minne för att skicka och ta emot data från Proxy:n. Exempelvis så använder sig Automotive Service av Binder IPC för att kommunicera med applikationers Automotive Managers.

SDP

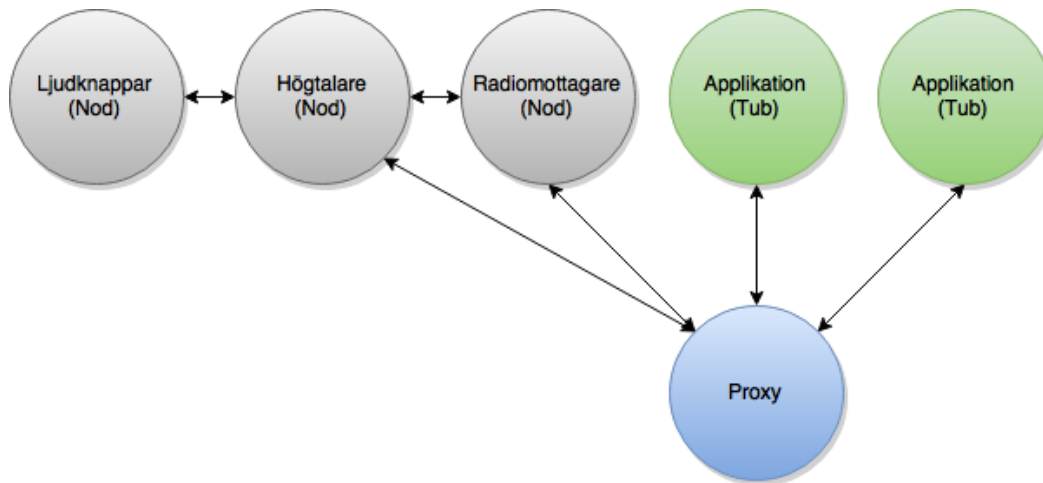
Inuti AGA-nätverket så är kommunikationen baserad på metदानrop och SDP-meddelanden. Proxy:n är en öppen nod som applikationer kan ansluta till, även en OEM kan ansluta genom VIL-lagret.



Figur 6: Kommunikationsmetoder i AGA [1]

3.9 Topologi

AGA är ett mesh-nätverk på grund av att alla SDP-noder i AGA har möjligheten att skapa AGA-gateways som tillåter dem att kommunicera direkt med varandra samt med noder som är utanför AGA-enheten, som i en separat ECU. Figur 7 visar hur ett typiskt meshnätverk i AGA kan se ut, de flesta noder är förmodligen kopplade direkt till proxy-noden. Vissa noder är även ihopkopplade med varandra då det tillåter för snabbare kommunikation. Vissa noder ligger utanför AGA-enheten och är bara sammankopplade med en nod i AGA-enheten utan att vara ansluten till proxy-noden. Däremot så är alla tuber direkt anslutna till proxy-noden.



Figur 7: Ett mesh-nätverk i AGA

4 Gränssnitt för kommunikation

Detta avsnitt presenterar de gränssnitt och protokoll som en bil kan använda sig av för att kommunicera med sin omvärld och ett antal svagheter däri. Kapitlet är uppdelat i två delar, den första delen beskriver de fysiska medier som kan utnyttjas för att kommunicera med fordonet, den andra delen beskriver de protokoll som kan användas för kommunikation.

4.1 Gränssnitt mot fordonet

Gränssnitten mot fordonet syftar på de fysiska medier som förbinder en enhet med fordonet och möjliggör kommunikation. Gränssnitt bildar attackytor eftersom de kan utnyttjas för att utföra intrångsförsök i systemet. Inför denna rapport har vi inte haft tillgång till att testa dessa gränssnitt då det är tänkt att de ska implementeras av OEM:er.

4.1.1 Bluetooth

Bluetooth möjliggör för trådlös kommunikation mellan mobila enheter över korta avstånd. Med hjälp av en antenn kan denna räckvidd utökas [11]. Detta är användbart i fordon för att skicka och ta emot data mellan enheter trådlöst. Det har blivit populärt bland fordonstillverkare att låta användaren koppla sin mobiltelefon till sitt fordon med hjälp av Bluetooth för att bl.a. förenkla telekommunikation med andra individer. På grund av sårbarheter i Bluetooth som kan utnyttjas av t.ex. mobila enheter för att möjliggöra obehörig tillgång till användarens system blir därför Bluetooth en möjlig attackyta för angripare. En angripare kan söka efter och lokalisera Bluetooth-enheter inom dess räckvidd för att sedan utnyttja svagheter i Bluetooth för att t.ex. få tag på känslig information från dessa enheter. Ett exempel på en känd svaghet är CVE-2011-4276 som i tidigare Android, version 2.3.0 upp till 2.3.6. tillåter angripare inom Bluetooths räckvidd att ta emot kontaktdata via enhetens telefonbok [12].

Bluetooth i fordon

Även användarens fordon är sårbara för attacker som använder sig av Bluetooth. En applikation som utnyttjar en känd brist för Bluetooth i fordon är "the car whisperer", som utvecklades av en grupp kallad Trifinite. The Car Whisperer utvecklades för att visa på bristerna i de Bluetooth-system som finns i vissa fordon. Car Whisperer utnyttjar svagheten att lösenorden för vissa hands-free system oftast består av en fyra-siffrig säkerhetskod (t.ex. 1234) för att tillåta en enhet att ansluta sig till systemet. Enheten som ansluts får tillgång till att ta emot och skicka ljud till fordonet [11].

4.1.2 4G

4G är fjärde generationens mobila telekommunikations teknologi. Det som gör 4G unikt är att den är helt IP-baserat, och transporterar trafiken direkt från en mobil enhet till en annan

genom en trådlös kopplingspunkt. 4G har under den senaste tiden blivit väldigt intressant för fordonsindustrin. Denna uppkoppling skulle möjliggöra flera funktioner och förbättringar för användaren som exempelvis tillgång till bättre internetuppkoppling och tillåta en förenklad distribution av mjukvaruuppdateringar från tillverkaren [13].

Problematiken med 4G är att den är sårbar mot enkla störningstekniker som blockerar dess signal. Dessa störningar kan utföras med en sändare som riktas mot delar av 4G signalen för att slå ut dess basstation. Utrustningen får plats i en portfölj, och kan slå ut flera kilometer av 4G täckning [14].

4.1.3 USB

USB är ett snabbt och effektivt sätt för att skicka och ta emot data mellan två enheter. USB är användbart i ett fordon då det kan användas för att ge användaren möjlighet att koppla in en mobil eller ett USB-minne för att spela upp musik eller för att tillåta en reparatör att uppdatera fordonets mjukvara.

Bad USB är en programvara som kan installeras på USB-enheter. Denna skadliga programvara kan få otillåten tillgång till USB-enhetens värd, till exempel en dator, utan att användaren vet om det. För att programvaran ska få kontroll över USB-enheten måste ett chip i USB:n kallat controller chip omprogrammeras. En sådan omprogrammering utgör ett stort problem då det oftast inte finns något skydd mot dessa. Efter att enheten är omprogrammerad kan den överta dess värd på många olika sätt. Ett exempel är att den infekterade enheten kan emulera ett tangentbord och på så sätt utföra kommandon som den inloggade användaren. På detta sätt kan den infekterade enheten även fortsätta attacken genom att infektera andra USB-enheter som är inkopplade i värd-datorn. I dagens läge finns det tyvärr ingen lösning som används för att skydda sin USB-enhet från detta problem. En möjlig lösning skulle kunna vara att företagen implementerar kryptering på deras controller chip som tar bort möjligheten att omprogrammera dem [15].

4.1.4 AUX

En AUX-port, Auxiliary port, är en asynkron serieport. AUX är ett gränssnitt som tillåter en enhet att ta emot och skicka ljudsignaler och används ofta i ljudutrustning som t.ex. högtalare [16].

IT-företaget Cisco upptäckte att deras AUX-port innehöll en sårbarhet i deras IP-telefoner; genom att koppla in sig via AUX-porten på baksidan av enheten var det möjligt att få lokal tillgång och på så sätt injicera skadlig kod [17].

4.1.5 CD-spelare

CD-spelare är en väl använd teknologi som finns i fordon och används främst för att spela upp ljudfiler som laddas in från en cd-skiva men kan även användas för att uppdatera mjukvara i vissa fordon. Problemet med vissa CD-spelare är att deras mjukvara kan innehålla buggar som kan göra det möjligt för en cd-spelare att läsa av skivor med filer som utnyttjar dessa buggar. En skadlig fil på cd-skivan kan fungera och se ut som en vanlig ljud-fil men egentligen utföra en attack mot fordonet utan användarens vetskap[18]. I dagens läge verkar det dock finnas ett minskat intresse av att ha CD-spelare i fordon. Det är möjligt att de inte längre kommer att finnas som standard i fordon inom några år.

4.1.6 WiFi

WiFi är en teknik för att skapa trådlösa nätverk. Dess syfte är att skapa ett nätverk som kan koppla samman flera enheter trådlöst. WiFi i fordon är tänkt för att låta användaren få tillgång till bättre internetuppkoppling för sina enheter och för vissa fordon är det även tänkt att kunna få mjukvaruuppdateringar från tillverkaren genom denna uppkoppling [19].

4.1.7 HMI

HMI, Human Machine Interface, är panelen i förarutrymmet, som gör det möjligt för kommunikation mellan användaren och fordonet. Inom ett fordon innehåller benämningen kopplingen mellan fordonet och omvärlden.

4.2 Protokoll

Protokollen nedan använder olika sätt för att transportera eller hanterar data över de fysiska medierna. Dessa gränssnitt skulle kunna utnyttjas för en attack mot fordonet och utgör därför ett hot. De digitala gränssnitten har funnits tillgängliga inom ramen för detta projekt och har därför varit möjliga att testas och analyseras.

4.2.1 TCP/IP

TCP/IP används mellan SDP-noder för att skicka SDP meddelanden mellan enheter som är fysiskt åtskilda. SDP kör över TCP som visas i figur 8, och tillåter adressering som är standardiserad mellan enheter. Då meddelandet passerar in i en AGA-enhet översätts det tillbaka till ett SDP-meddelande och är redo att sändas ut i AGA-nätverket.



Figur 8: TCP och SDPs förhållande i kommunikationsstacken

TCP problem

TCP är ett transportprotokoll och är inte avsedd att skydda trafik. Detta kan utvecklare gå miste om när de använder protokollet vilket kan orsaka problem som säkerhetsbrister. Ett exempel på attacker som TCP/IP kan utsättas för är sniffning och spoofing. Paket-sniffning är en attack som sker genom avlyssning av trafiken som sänds över en delad nätverkskanal. Flera protokoll som körs över TCP, såsom FTP eller Telnet, är sårbara mot paket-sniffning för avlyssning av känslig information [20]. Ett sätt att öka säkerheten och skydda trafiken är att använda IPSEC/SSL/TLS som ger ett extra protokoll-lager där fokus har lagts på säkerhet[21].

IP spoofing utnyttjar tilliten mellan två värdar. Angriparen skapar ett IP-paket med en falsk källadress vilket bedrar programmet vars autentisering förlitar sig på användarens källadress som identifierare. Detta möjliggör otillåten access till systemet [22].

4.2.2 SDP svagheter

Eftersom SDP är ett nyutvecklat protokoll för AGA (se 3.7) som utvecklats utan ett säkerhetsperspektiv finns det risk att SDP innehåller buggar. En bugg i detta protokoll skulle möjligtvis kunna utnyttjas för ett intrång i AGA.

4.2.3 Protokoll för användarapplikationer

Android applikationer innebär en stor säkerhetsrisk eftersom de ofta har en bred attackyta för Android-enheter. Avsiktligt eller oavsiktligt kan de installeras på enheten och kan utgöra ett stort hot då de på olika sätt kan injicera skadlig kod. Dessa applikationer kan uppföra sig som en vanlig applikation men utan att användaren vet om det utföra oönskade aktiviteter såsom spara känslig användardata.

Webbläsaren

Webbläsare är en programvara som tillåter en enhet att besöka webbsidor, den gör det möjligt för en enhet att läsa till exempel HTML-kodade dokument som hämtas från webbservrar [23]. En Webbsida kan utnyttja brister eller sårbarheter i en webbläsare för att exekvera skadlig kod.

Skadliga applikationer

Skadliga applikationer är applikationer som utnyttjar sårbarheter och brister för att få tillgång att utföra obehöriga aktiviteter på enheten. I dagsläget existerar ett stort antal Android-applikationer som innehåller skadlig programvara. Ett exempel på denna typ av skadlig programvara är Fitikser som är en trojansk häst. Utan användarens vetskap stjälar Fitikser information som till exempel telefonsamtal och sms [24].

Buggar

Applikationer innehåller ofta buggar. Mestadels är det applikationer som utvecklats utan en bra säkerhetsstruktur, eller inte blivit tillräckligt testade, som kan innehålla brister som äventyrar säkerheten. Ett exempel kan vara att en applikation kan ge obehöriga fel privilegier eller t.o.m. få Android enheten att göra en soft-reboot. Buggarna kan ge olika konsekvenser, allt ifrån crashar av Android-enheten till värre där en angripare kan exkavera kod eller på annat sätt utnyttja buggen för att utföra sin attack.

En applikation som saknar en bra säkerhetsstruktur kan vara mycket sårbar mot attacker. Om applikationen sänder okrypterad känslig information mellan två noder över ett nätverk kan den "sniffas" upp av en angripare med verktyg som Wireshark. Dessa meddelanden kan sparas och modifieras för att sedan skickas igen till den andra noden utan någon av nodernas vetskap, detta utgör en risk för man-in-the-middle och replay-attacker.

BMW's connected drive innehöll en liknande sårbarhet. I Connected Drive finns en funktion som tillåter en användare som låst ute sig ur sitt fordon att öppna den genom att kontakta BMW's assistans-linje, som sedan kan öppna fordonet på distans. En tysk fordonsgrupp, som kallas Allgemeiner Deutscher Automobil-Club (ADAC), fann genom en analys av denna mjukvara att det gick att återuppspela meddelanden och på så sätt kunna utnyttja funktionen på samma sätt som BMW's assistans linje kunde göra. Sårbarheten innebar att det var möjligt för en obehörig att öppna fordonet genom att utnyttja buggen. [25].

5 Metod

I detta kapitel presenteras en redogörelse för hur datasinsamlingen har skett och de metoder som har använts under arbetet, intervjuer och fallstudier. Kapitlet går sedan in djupare på fallstudien och hur denna har konstruerats och sedan på de verktyg som har använts under arbetets gång.

5.1 Metoder för datainsamling

Följande metoder har använts i examensarbetet:

1. Arbetet inleddes med instudering av koden och arkitekturen eftersom det ansågs vara en grund för att föra intervjuer.
2. Därefter hölls intervjuer med utvecklare följt av datasäkerhetsexperter och till sist OEM-tillverkare för att få en förståelse för tankesätt, svagheter och framtiden för AGA. Dessa intervjuer bedrevs med målet att ta fram hypoteser om de svagheter som existerar i AGA.
3. Sedan gjordes en fallstudie där de hypoteser som intervjuerna gav upphov till testades och de som troligen kunde påvisas kartlades vilket gjorde det möjligt att formulera lösningar och skäl till dessa.

5.1.1 Instudering

För att de andra metoderna för informationsinsamling skulle resultera i meningsfull data så var det första steget en instuderingsperiod där all dokumentation och även koden studerades. Detta var för att kunna diskutera och för att ha förståelse på likvärdig nivå som de intervjuade.

5.1.2 Personliga intervjuer

Inför intervjuerna så valdes en öppen intervjuform som var icke-standardiserad framför en styrd eller standardiserad för att låta de intervjuade att få fram sina egna idéer och tankar [7]. För att få en bild av AGA så intervjuades Christopher Pavlic, utvecklare och Mohammadreza Javaheri som är projektledare för utvecklingsgruppen på Combitech och för att få information från ett OEM-perspektiv så intervjuades ledande personer inom AGA på Volvo. Vi har även haft en intervju med Frederik Holgersson och Caroline Bildsten (som är Combitechs expert på Androidsäkerhet), samt ett tiotal andra. Där igenom så kunde information från experter med stor erfarenhet inom kod och applikationsgranskning inhämtas. Då arbetet ännu inte var redo för en fullständig utvärdering gav intervjuer en riktning till arbetet mot de problem i AGA som var troliga.

5.1.3 Fallstudie

Då fallstudier bedrivs med informationsorienterade tester [9] så studerades de reaktioner som AGA har mot olika typer av attacker. Sedan valdes ett par fall där reaktionerna ledde till svagheter i systemet varefter arbetet fortskred från den punkten. Fallstudien var lämplig för

projektet då målet var att finna och studera svagheter i AGA, vilket kan definieras som oväntade fall i AGA från utsida stimulans. Därefter kunde det undersökas vad som orsakade reaktionen och varför vilket i sin tur gjorde att en lösning kunde designas baserat på det.

5.2 Design av experiment

5.2.1 Design av intervjuer

Intervjuerna planerades sådana att intervjuaren, i de flesta fall båda projektmedlemmarna, gick in med en väl förberedd grund med ett underlag som fick leda intervjuerna men att den var såpass öppen att ingen information skulle missas. Bland annat så studerades arkitekturen i detalj innan intervjuer med utvecklare och frågor förbereddes mot troliga svagheter i den. Även koden och kodstrukturen blev studerad innan arbetet började för att ge en grund till intervjuerna. Allt detta gjorde även att intervjuaren var tillräckligt väl införstådd i projektet för att svaren skulle ge meningsfulla resultat i tal mot utvecklare men också att intervjuaren skulle kunna förklara AGA grundligt nog för en utomstående säkerhetsexpert för att även där ge meningsfulla svar.

5.2.2 Design av fallstudier

Fallstudierna baserades på de svar som gavs under intervjuerna och i samråd med säkerhetsexperter som gav oss vägledning i hur de olika svagheterna skulle kunna testas. De svagaste gränssnitten valdes ut för attacker och reaktionerna studerades och blev indelade i fall. Ett basfall, vilket beskrivs nedan, jämfördes mot andra fall. Sedan undersöktes och dokumenterades skäl till det beteende som observerades. Utstickande fall var bland annat ett sätt att otillåtet läsa och skriva meddelanden i systemet eller i värsta fall på ett replikerbart sätt forcera en omstart av Android.

5.2.3 Basfall

AGA har en proxysserver som kan leverera SDP-meddelanden till olika noder. Dessa SDP-meddelanden kan bara skickas av behöriga noder eller en simulator och samtidigt endast tolkas av dessa.

5.3 Verktyg

5.3.1 Kodhanteringsverktyg

Android studio har använts för att utveckla ett program som riktar sig mot svagheter i AGA-arkitekturen med SDP. Programmet har skrivits som en Androidapplikation för att installeras på en surfplatta.

5.3.2 Säkerhetsverktyg

För att undersöka de öppna nätverksportarna i AGA har vi använt **Nmap**, en gratis portscanner, som används för att upptäcka tjänster i ett datanätverk. Som komplement till Nmap har **OpenVAS**, ett mer fullskaligt datanätverkssäkerhetsverktyg använts.

Scapy är ett mer nätverkspaketsmanipulationsverktyg som tillåter användaren att modifiera eller skapa paketen och skicka dessa. Den har fler funktioner för att modifiera paketen än Nmap. Scapy har använts för att undersöka AGA genom att skapa och återuppspela nätverkspaket. Även **TCP-replay** har körts för att återuppspela fångade paket. **Wireshark** har använts för att studera och isolera relevant nätverkstrafik.

Understand som är utvecklat av Scitools har använts för viss kodgranskning i delar där svagheter var hittade för att samla in information för säkerhetslösningar.

5.3.3 Övriga

Zbee är ett eldrivet testfordon som användes för testning.

6 Genomförande

6.1 Uppstart

I den inledande fasen bedrevs instudering och uppsättning av miljöer och program, instuderingsfasen beskrivs i 4.1.1 och de verktyg som har använts i 4.3.

6.2 Inledande intervjuer

De inledande intervjuerna bedrevs med utvecklare, OEM och en säkerhetsexpert för att få en utgångspunkt i arbetet. Vi fokuserade på arkitekturen, framtiden för AGA och en utvärdering av dessa med vår säkerhetsexpert. Efter intervjuerna så samlades all data på förslag av vår handledare till en tabell över alla gränssnitt tillsammans med deras troliga svagheter. Därefter valdes de mest troliga och farligaste svagheter ut för att undersökas. Vi hade mest information om arkitekturella svagheter och började således med dessa.

6.3 Fallstudie med arkitekturella svagheter

Vi identifierade tre ursprungliga intrångsvägar genom arkitekturen: förbikoppling av säkerhetsfunktioner, certifikat-spoofing och att kringgå policy manager:n. Dessvärre var certifikat och policy manager endast implementerade som en skelettversion med tanken att låta OEM implementera sin egen version. Detta lämnade en intrångsväg att testa, förbikoppling, alltså om det går att koppla in sig i AGA-nätverket och kringgå säkerhetsfunktioner. Vi uppnådde detta genom att skapa en applikation som i sin tur skapar två SDP-noder, utan att använda AGAs normala funktioner, en som lyssnar på alla signaler och en som kan skicka alla signaler. Noderna kopplar in sig till AGA-nätverket på port 8251, vilket är porten som hanterar meddelanden i AGA.

6.4 Uppföljande intervjuer

Målet med nästa runda av intervjuer var att konfirmera och diskutera det som vi hade hittat samt i samråd med experter bestämma nästa målsättning. Vi bedrev dessa intervjuer med en vidare krets av säkerhetspersonal, kodgranskare och Androidspecialister samt penetrationstestningsexperter. Deras unika kunskaper hjälpte oss att hitta andra attackområden i AGA. För att få ytterligare feedback så hade vi också intervjuer med utvecklare där vi diskuterade våra resultat. Vi fick även ett förslag från utvecklingsteamet att studera en hypervisorlösning för AGA som vi började undersöka och även diskuterade med vår handledare. I denna fas förfinade vi vår tabell.

6.5 Fallstudie med nätverkssvagheter

Då AGA kommunicerar bland annat genom TCP/IP så ville vi studera om vi kunde hitta några nätverkssvagheter i AGAs TCP/IP-kommunikation. Vi började med att testa om det fanns fler öppna portar än förväntat på den enhet som kör AGA. Det första verktyget som användes var Nmap men när vi testade med Nmap mot AGA-enheter för att upptäcka eventuella öppna

portar så upptäckte vi att det är möjligt att få system-tjänsten att stanna och på så sätt få surfplattan att göra en soft reboot. Detta noterades som en avvikelse från basfallet och det bestämdes att vi ville studera denna i sitt eget skede. Vi fortsatte sedan med att använda Scapy, TCP-Replay och Wireshark för att återanvända skickade paket på nätverket.

6.6 Fallstudie om reboot

Vi ville studera vad exakt det var som fick AGA-enheten att genomföra en soft reboot d.v.s. en omstart. Vi studerade den platsen i koden som tar hand om nätverkstrafik och även hur paketen ser ut som får plattan att göra en omstart. Vi använde Wireshark för att spela in paketen och studera dessa, sedan användes Scapy för att skicka paket på samma form för att se när AGA-enheten gör en omstart. Samtidigt så använde vi ADB, Android Debug Bridge, för att läsa logmeddelandena från surfplattan vilket gjorde att vi kunde hitta vad i koden som orsakar en omstart, och varför, för att dubbelt styrka skälet till omstarten (se 7.1.2).

6.7 Ta fram förslag

Efter alla fallstudier var färdiga togs två förslag mot olika delar av programmet fram. En som skulle skydda AGA-nätverket mot en angripare som angriper från AGA-enheten och en som skulle skydda fordonet mot en angripare som har kontroll över AGA.

6.8 Avslutande presentationer

Avslutningsvis ville vi förmedla ut våra tankar och förslag till de som varit involverade i vårt arbete. Därför hade vi en presentation på Combitech om vår analys och de tankar som vi hade om projektet. Vi hade även presentationer för säkerhetskollegor och utvecklare där vi fick respons på våra lösningar.

Mot slutet av projektet fick vi även möjligheten att få intervjua en av utvecklarna till Mentor Graphics inbyggda hypervisor. Denna intervju gav understöd till de teorier vi har om hypervisors (se 8.1).

7 Resultat

I detta kapitel så presenteras resultaten på de frågeställningar som är definierade i kapitel 1.4 i inledningen; vilka brister existerar och hur de kan utnyttjas (se 7.1), vilka säkerhetsmekanismer existerar (se 7.2) och vilka säkerhetsmekanismer bör implementeras och hur (se 7.3). I Tabell 1 finns en kort översikt över svaren på fråga 1 och 3.

Tabell 1: De brister som är identifierade och deras lösningar.

Brist	Effekt	Lösning
Fordonet är ej skyddat mot ett komprometterat AGA.	Fordonet är mycket sårbart mot ett AGA med virus eller liknande.	Gateway (Avsnitt 7.3.1)
Icke fångat undantag	Det går att få AGA att crasha.	Hantera det missade undantaget
Informationssårbarhet mellan nätverk.	Det går att avlyssna på kommunikationen mellan nätverk.	Kryptering & Signering (Avsnitt 7.3.2)
SDP DoS	Det går att med en falsk nod neka AGA att använda valfria signaler.	Autentisering (Avsnitt 7.3.3), IDS (Avsnitt 7.3.6) och att lösa buggen med SDP-meddelanden.
SDP Replay	Det går att fånga in och skicka information igen, alltså skicka signaler till AGA från ett annat nätverk.	Autentisering (Avsnitt 7.3.3) och Kryptering & Signering (Avsnitt 7.3.2)

Det identifierades fem allvarliga brister som har fått lösningsförslag. Övergripande så anser vi att AGA behöver använda kryptering och vi tror att det skulle lösa flera av de brister som vi har hittat men också sådana som vi inte har hittat. En annan punkt som är viktig är fordonets isolering från AGA vilket vi går vidare in på i kapitel 8.1.

7.1 Identifierade brister och hur de kan utnyttjas

En kartläggning av gränssnitt presenterades i kapitel 4. Efter evaluering av protokollen så har flera buggar och sårbarheter hittats. Vissa av dessa härleder från arkitekturen och andra från koden. Lösningar för dessa diskuteras i 7.3, 8.1 och 8.2.

7.1.1 Fordonet är ej skyddat mot ett komprometterat AGA

AGA är byggt på Android där mycket makt ligger hos användaren. Ofta är virus gömda i applikationer som begär udda tillåtelser vilka oftast tillåts av användaren [26], som Durak [27] vilket gömde sig i ett patiens-kortspel. På grund av Google Plays öppna inställning till applikationer är Android ofta en enkel måltavla för angripare. Detta blir ett mindre problem om det finns en skraddarsydd marketplace skapad av en OEM/fordonstillverkare som

undersöker och certifierar applikationer innan de läggs till i marketplace [47]. Med de identifierade sårbarheterna räcker det inte att lösa problemen och skydda själva programmet, utan för att säkerhetskritiska funktioner inte ska störas behöver även fordonet ett skyddslager mot AGA. En annan svårighet som kan ge upphov till något liknande är uppdateringscykler; i nuläget är det oklart vem som ansvarar för att AGA ska hållas uppdaterat. Kanske kommer AGA att även existera på flera olika hårdvaror, i flera olika biltillverkares bilar. Detta kan leda till att AGA innehåller säkerhetshål som är kända och har patchats i nyare versioner av Android.

7.1.2 Icke infångat undantag

Då en öppen port (8251,9898,9899) mottar ett nytt uppkopplingsförsök, som inte fullföljs så har AGA ingen funktionalitet för att hantera den avbrutna uppkopplingsförsöket. Trots den avbrutna uppkopplingen försöker AGA skicka ett `sendGreeting` meddelande, vilket resulterar i ett ohanterat undantag vilket leder till att AGAs Automotive Service crashar och därför genererar en omstart. Efter granskning av kod och felmeddelanden stod det klart att det var funktionen `sendGreeting` i klassen `AbstractEthGatewayNode.java` (se bilaga A) som innehöll felet. Funktionen `sendGreeting` skickar iväg en "hälsning" till klienten som den sedan väntar på svar ifrån. Problemet är att `sendGreeting` förlitar sig på att uppkopplingen till klienten är uppe. Om uppkopplingen mot klienten stängs av klienten så kastar `sendGreeting` ett undantag upp till funktionen som kallade på `sendGreeting`. Denna tas dock inte omhand i funktionen som kallade på `sendGreeting`. En avbruten uppkoppling går att generera med ett flertal verktyg, Nmap, OpenVAS eller Scapy och utgör en konstant attackpunkt mot en AGA-enhet. Alltså kan denna omstart göras från en enhet med nätverksåtkomst mot en uppkopplad AGA-enhet när som helst och det enda kravet är att känna till IP-adressen. Lösningen för det icke infångade undantaget var att implementera kod i de kallande funktionerna som tar emot det och lägger ut ett felmeddelande till användaren.

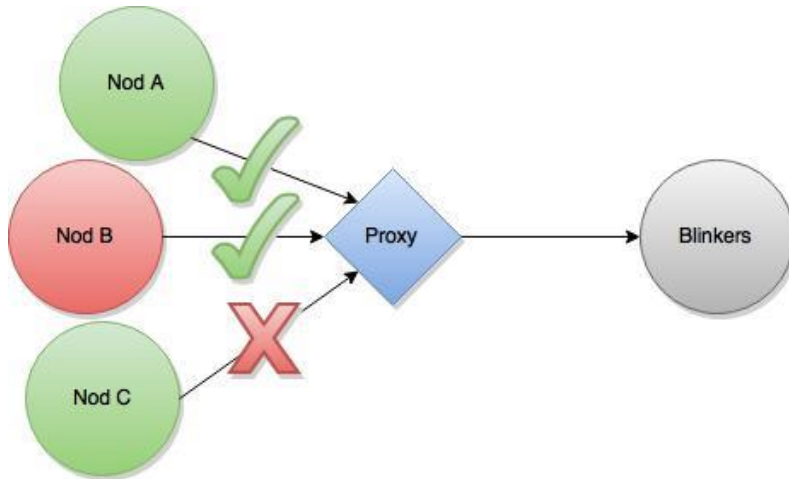
7.1.3 Informationssårbarhet mellan nätverk

Då SDP-meddelanden färdas mellan gateways befinner de sig okrypterade i TCP och CAN. Detta leder till att det går att läsa och ta del av den information som finns i dessa meddelanden. Sårbarheten sträcker sig även till replay-attacker då paket kan spelas in och spelas upp senare för att uppnå något mål, som till exempel att visa upp fel hastighet i hastighetsvisaren eller om AGA får sådana befogenheter att aktivera blinkers. Det är dock inte troligt att AGA får ta hand om kritiska funktioner så en angripare kommer inte att kunna bromsa eller öka farten. Med tillräcklig kunskap är det troligt att dessa paket även kan skapas och skickas in i AGA-nätverk där de kan påverka viktiga funktioner.

7.1.4 SDP DoS

Då ett stort antal meddelanden med samma signal-ID skickas från en SDP-nod kan signalhanteringen till slut blockeras och det signal-ID:t inte längre skickas. Detta härrör från

en bugg i AGA där ett speciellt mönster krävs mellan subscribers och providers för att det ska vara klart vilken nod som får skicka en signal och till vem. Resultatet blir att det går att blockera viktig information från att skickas till föraren under tiden fordonet körs. Till exempel, på AGA-prototypfordonet går det att inaktivera kommandot för att aktivera blinkers.



Figur 9: Noder försöker kommunicera med blinkers

I fallet i figur 9 så vill Nod A och C skicka signalen för blinkers. Detta fungerar för nod A men när nod B som är en elakartad nod skickar samma signal upprepade gånger så blockeras Proxy:n som slutar acceptera blinkers-signalen vilket gör att när nod C sedan försöker skicka signalen blir den nekad. Signalföljden kan studeras i Tabell 2. Blinkersen går nu ej att aktivera. Detta skulle vara extra farligt om endast de yttre blinkerssignalerna blir avaktiverade medan de inre fortfarande säger till föraren att de är aktiva.

Tabell 2: Signalföljd för noderna A,B & C.

Proxy	Nod A	Nod B	Nod C
OK	Jag vill skicka signalen Blinkers		
OK	Blinkers: PÅ		
OK		Jag vill skicka signalen Blinkers	
OK		Blinkers: AV	
OK?		Jag vill skicka signalen Blinkers	
FAIL		Blinkers: AV Och så vidare...	
FAIL			Jag vill skicka signalen Blinkers
FAIL			Blinkers: PÅ

7.1.5 SDP Replay

En signal kan nu spelas in när den är paketerad i IP-paket och skickas igen för att senare plockas upp av AGA-nätverket som en riktig signal. Detta leder från ett signalhanteringsproblem då det inte finns något sätt att verifiera att ett paket är korrekt.

7.2 Existerande säkerhetsmekanismer och hur de kan förbättras

7.2.1 Policy manager

AGA:s policy manager förlitar sig på att OEM:en bestämmer vilka signaler som är tillåtna eller ej. Policy managern är lämnad oimplementerad och det är meningen att en OEM-tillverkare skall implementera den. Policy manager:n skulle fungera bättre om den undersökte alla skickade signaler och inte bara de signaler från SDP-noder som representerar applikationer. Det betyder att alla noder, och inte endast de som representerar applikationer, skulle ligga i tuber. I nuläget så kan en SDP-nod som är inkopplad i AGA-nätverket skicka och ta emot valfria signaler oavsett om den är den legitima avsändaren eller mottagaren för dessa. Vi skulle gärna se en mycket striktare signalhantering där endast de signaler som är viktiga för noder kan hanteras av respektive nod; att en signal om ökad ljudnivå ska gå till högtalarna är okej, men det finns inget skäl för solluckan att hantera den.

7.2.2 Robusthet i AGA:s arkitektur

AGA-arkitekturen i sig leder till vad man skulle kunna kalla en säkerhetsmekanism. Då SDP, som diskuteras i 3.7, är ett nytt slags kommunikationsprotokoll med sin egen nätverksstruktur så finns det ännu inga kända exploits mot AGA. Att då hitta och skriva en ny exploit som inte bara riktar in sig på Android utan även mot AGA kräver en viss expertis vilket utesluter mindre avancerade angripare. Sättet SDP är uppbyggt på, med providers och subscribers gör att man inte kan sända ett meddelande till en speciell nod utan att den själv lyssnar efter meddelanden med det ID som angriparen skickar. Eftersom det inte går att få tag på en lista över subscribers går det inte heller att veta slutdestinationen för ett speciellt paket vilket gör det svårt att rikta sig mot en specifik nod.

7.3 Lämpliga säkerhetsmekanismer och hur dessa skall implementeras

7.3.1 Gateway

Det skulle vara förmånligt att lägga till ett lager av separering mellan AGA och CAN-bussen. Vi förespråkar en slags brandvägg som kan skydda fordonet mot skadlig trafik från AGA. Vi beskriver först en gateway i mjukvara och sedan en i hårdvara.

Gateway i mjukvara

En gateway som ligger i en separat virtualiserad maskin på samma enhet och som kan filtrera skadliga och oskadliga meddelanden från AGA skulle ge fordonet ett extra lager av skydd

mot ett komprometterat AGA. Att använda en hypervisor, en virtualiseringsmjukvara som tillåter flera operativsystem att köra samtidigt på samma maskin [28], skulle tillåta att viktiga systemfunktioner endast körs på det ena virtuella systemet. En virtuell gateway skulle inte öka hårdvarukomplexiteten i fordonet utan kunna köras på samma hårdvara, men om enheten kan styra kritiska funktioner som acceleration eller bromsar så är bedömningen att det inte är säkert nog med endast en mjukvarugateway. Ett av operativsystemen som körs skulle vara AGA som bidrar med infotainment, det andra ett RTOS, ett realtidsoperativsystem som har möjligheten att kommunicera med ett fordon. Förslagsvis skulle detta RTOS vara ett pålitligt OS som är specialiserat för en fordonsmiljö. Om dessa operativsystem sedan delas genom en hypervisor så skulle en säkrare uppdelning kunna uppnås. I Tabell 3 återfinns en lista över vilka funktioner som körs på respektive operativsystem.

Tabell 3: Översikt funktionsdelning

RTOS	AGA
Kommunikation mot CAN-bussen	Infotainment
Back/fram kamera	UI
Klimatkontroll	3G/4G, Bluetooth, WIFI
Gateway	GPS
	CD, AUX, USB

Vi ökar pålitligheten genom att vi flyttar funktioner som manipulerar fordonet till RTOS:et medan vi lämnar infotainmentfunktioner i AGA. Dessutom läggs det till ett lager av säkerhet. Genom att använda en hypervisor för att hantera informationen som flödar mellan AGA och RTOS så skärmas AGA av från omvärlden och måste gå genom en Gateway i RTOS som kan säkerställa de signaler som AGA skickar; att hämta information om hastigheten är tillåtet men att öka den är förbjudet. Det skulle fortfarande vara acceptabelt att låta AGA skicka förfrågningar om t.ex. hastighet till RTOS om de undersöks av en Gateway för suspekt trafik. Hur den ska tolkas görs förslagsvis genom någon tröskel för normal trafik men lämnas till en OEM för implementation.

Säkerhetsrisker vid användning av en hypervisor

Att använda en hypervisor för att virtualisera AGA och ett RTOS leder till vissa säkerhetsproblem. Hypervisorn är högintressanta för angripare då de har tillgång till känsliga system på dess värdsystem. En väg för angriparen att försöka nå hypervisorn är genom den virtuella maskinen. En känd attack är hyperjacking. Hyperjacking är en attack där användaren

tar kontroll över en hypervisor [29]. Detta burkar ofta ske genom en hypervisor escape. Med hypervisor escape menas att man bryter sig ut från den virtuella maskinen till värdens operativsystem. Detta kan ge angriparen administrativa rättigheter och även tillåta vederbörande att köra kod [30]. I vårt system så skulle detta kunna betyda att en attack bryter sig ut från AGA för att få möjlighet att anfälla systemets RTOS.

Hypervisors kan användas för rootkit attacker, vars uppgift är att dölja sig eller andra specifika processer och program som utförs på datorn [31]. Ett exempel på ett rootkit är Blue Pill. Tanken bakom detta är att den ska utan användarens vetskap starta en tunn hypervisor som virtualiserar hela datorn. Operativsystemet skulle fungera som vanligt men Blue Pill skulle få tillgång till i stort alla funktioner på datorn. Detta skulle kunna utföras på vårt system [32].

Gateway i hårdvara

En hårdvarugateway kommer att vara isolerad från fel i AGA och kommer att kunna skydda fordonet även om AGA skulle bli hackad eftersom den skulle vara skyddad mot en illasinnad uppdatering av dess firmware. Den skulle ha sin egen specialiserade hårdvara och vara kopplad mellan AGA och fordonet för att ha förmågan att tolka och filtrera trafik från AGA på en låg nivå. Vi anser bestämt att en hårdvarugateway är nödvändig för att säkra ett fordon till den nivån att det är redo att köras i trafiken. Tabell 4 visar en sammanfattning av de två gatewaylösningar som beskrivits, där deras för- och nackdelar tas upp.

Tabell 4: sammanfattning av gatewaylösningar

Gateway i mjukvara	Gateway i hårdvara
Större attackyta	Mindre attackyta
Ökad komplexitet	Minskad komplexitet
Billigare än hårdvarulösning	Dyrare än mjukvarulösning

7.3.2 Kryptering & Signering

Att kryptera SDP-meddelanden skulle lösa problemen med konfidentialiteten i nätverket, paket skulle då inte kunna läsas av utomstående. Detta är en lösning till svagheten som diskuteras i avsnitt 7.1.2 och med hjälp av signering så skulle det även höja integriteten. Om man ytterligare inför strömkryptering vilket gör att meddelanden bara är giltiga en gång så stoppas även replayattacker som beskrivs i avsnitt 7.1.5. Ett förslag på en implementation av detta presenteras i avsnitt 8.2.

7.3.3 Nyckelhantering

Ett sätt att initiera nycklar till olika noder är att noderna skapar nycklar i ett tpm-chip[33] direkt i hårdvaran. Tyvärr så skulle då alla noder på en och samma enhet ha tillgång till alla dessa nycklar vilket gör att nycklarna inte kan användas för kommunikation inom den enheten utan bara är användbara vid kommunikation mellan enheter. Med hjälp av en public key infrastructure [34] kan noderna autentisera sig mot en proxynod som tillhandahåller sessionsnycklar vilka sedan kan användas för kommunikation med andra noder. En allvarlig säkerhetsbrist ligger dock i att AGA är baserat på Android som är ett OS med få säkerhetsfunktioner vilket gör att om flera noder ligger på samma hårdvara så kan en applikation med rooträttigheter läsa alla nycklar direkt från minnet. Ett säkrare sätt att tillhandahålla nycklar är att kompilera de individuella noderna med individuella konfigurationsfiler som innehåller nycklar vilket är sårbart om man kan få tag i bitkoden genom att till exempel ha tillgång till fordonet på något annat sätt men även genom säkerhetsbrister i Android, då nycklarna måste finnas i arbetsminnet för att användas. Dessa nycklar måste alltså vara unika per AGA-system och fordon för att höja säkerheten. Detta skulle vara samma modell som applikationer är byggda på med certifikat.

7.3.4 Autentisering

Då en nod önskar ansluta till AGA-nätverket så måste den ha ett giltigt certifikat signerat av en CA, certificate authority, och autentisera sig mot proxy-noden. Endast då ska den anslutande noden få tillgång till nycklarna som krypterar signalerna i AGA-nätverket.

7.3.5 Signalhantering

En nod ska ha en lista på signaler som den får skicka och lyssna på, att tillåta en nod som har hand om däcktrycket att till exempel skicka information om temperaturen i fordonet är felaktigt. Denna signalhantering skulle även kunna vara så finkornigt att en nod får skicka vissa värden av olika signaler men inte andra.

7.3.6 IDS

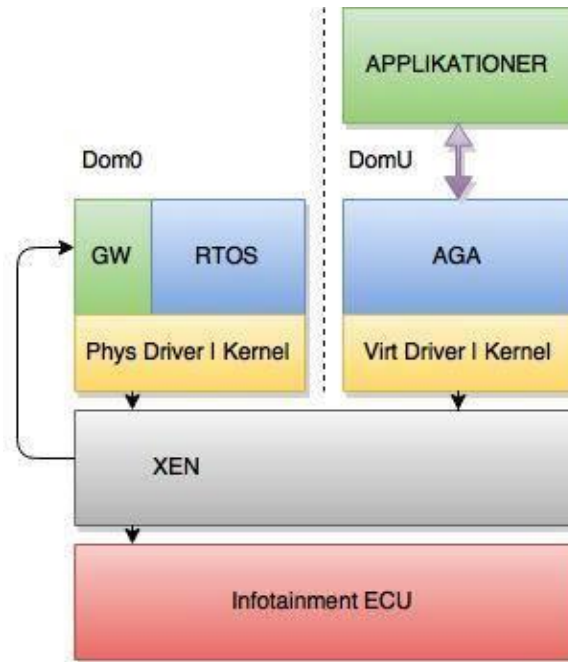
En IDS i AGA övervakar i bästa fall dynamiskt alla sända signaler och kan upptäcka om en nod uppför sig på ett oväntat sätt, till exempel genom att skicka ett flertal olagliga signaler eller på annat sätt bete sig på ett sätt som är ovanligt för nätverket. Den skulle då kunna varna föraren att ta in sitt fordon till en verkstad där tekniker skulle kunna ta hand om problemet.

8 Slutsats

Vi har hittat flera säkerhetshål och andra problem med AGA och även brist på implementation av säkerhetsmekanismer. Det går att bryta sig in på flera sätt och därför har vi tagit fram två förslag som skulle kunna vara möjliga för att förbättra säkerheten. Förslagen implementerar virtualisering och kryptering med de säkerhetsmekanismer som är beskrivna i avsnitt 7.3. Det första är ett skydd implementerat mellan AGA och hårdvaran för att minska påverkan på fordonet av ett infekterat AGA och den andra ett skydd implementerat för att minska chansen till att en angripare ska kunna manipulera AGA. Vi anser att båda är viktiga. Vi föreslår kryptering för att det är alldeles för lätt att idag läsa och manipulera meddelanden i nätverket. Ett annat förslag är en gateway mellan AGA och CAN-bussen för att bl.a. kunna flytta över säkerhetskritiska funktioner till ett säkrare operativsystem.

8.1 Förslag på implementering av virtuell gateway

Som det beskrivs i kapitel 7.3.1 så är det möjligt att ha en Gateway som separerats med hjälp av virtualisering. Det är ett viktigt steg att separera AGA från den miljö där en gateway körs. Genom att köra en så kallad bare metal hypervisor sätts hypervisorn upp direkt på hårdvaran utan ett annat operativsystem som hanterar den [36]. Detta ger hypervisorn direkt tillgång till hårdvaruresurser. Från detta kan flera operativsystem sättas upp i sina egna virtuella miljöer. Följande exempel är riktat mot en hypervisor vid namn Xen, vilken har en ARM-baserad AGA distribution som använder ARM Trust Zone för att dela upp hårdvara i säkra och normala världar. Den säkra världen kan endast ett operativsystem som klassats säkert utnyttja och en normal värld som alla operativsystem kan utnyttja. En Gateway ska vara specialiserad för skydda mot opålitlig AGA-specifik trafik. Ett RTOS som AUTOSAR skulle sedan genom Xen kommunicera direkt med bilen för att hämta information eller ge kommandon från AGA. Figur 10 är en schematisk bild över hur detta kan uppnås.



Figur 10: Översikt av hypervisorimplementation

Tabell 5: Fördelar och nackdelar med hypervisors

Fördelar	Nackdelar
RealtidsOS	Attackyta från AGA till hypervisor
Överflyttningen av säkerhetskritiska funktioner till ett säkrare OS.	Delat minne är inte säkert nog för att användas för att skydda säkerhetskritiska funktioner
Ett väldefinierat gränssnitt mot fordonet	Hypervisors har kända exploits

Som tabell 5 ovan visar har hypervisors fördelar som väldefinierade gränssnitt mot kommunikation med olika ECUer och även dess realtidskapabiliteter vilket gör det lättare att kommunicera med fordonet. Vi flyttar även över flera sårbara kritiska funktioner från AGA till den säkrare RTOS. En virtualisering öppnar dock upp möjligheter för attacker direkt mot

hypervisorn genom AGA, eller mot RTOS. Särskilt viktigt är realtidsfunktionaliteten då information i realtid är väldigt viktigt för vissa funktioner i ett fordon som till exempel hastighetsvisaren då den måste visa rätt hastighet vid rätt tidpunkt för fordonet, skulle denna information komma försent till föraren skulle det kunna få allvarliga konsekvenser.

Genom att dela upp systemet över två operativsystem så är det möjligt att höja säkerheten. Det finns olika sätt att göra detta på och de har både fördelar och nackdelar. Xen har valts då den har stöd för ARM som ofta används inom fordonsindustrin. [39] Xen har en låg kostnad prismässigt eftersom den är open-source och kan då även anpassas för specifika fordonslösningar. Lösningen gör dock att den totala attackytan skulle öka och möjligen leda fokuset av en angripare på sårbarheterna i hypervisorn. Xen innehåller flera brister och sårbarheter som påverkar säkerheten i värddatorerna. Guest to host escalation ger till exempel en angripare tillgång till administrativa rättigheter genom gäst-operativsystemet [43].

En annan möjlig lösning för att öka säkerheten i systemet är att implementera en hårdvarugateway. Denna lösning där man skiljer på operativsystemen i hårdvaran är mer tillförlitlig men den är även dyrare att implementera i större mängder. Utvecklare ställs mot en avvägning om vad som är rimligt mellan det ekonomiska och säkerheten. För ett företag som producerar färre enheter av en produkt kan en hårdvarulösning vara mer optimal då kostnaden för extra hårdvara inte blir särskilt stor under en mindre volym jämfört med en större. Däremot kan det vara mer lockande med virtualisering för ett företag med fokus på att massproducera sina enheter eftersom en mjukvarulösning blir billigare om man kan utnyttja samma hårdvara för flera ändamål.

8.2 Förslag på implementation av kryptering

Vi har valt att föreslå kryptering av AGA. Det skulle lösa flera av de problem som vi har identifierat som beskrivs i tidigare kapitel: replayattacker, svaghet över nätverk och införandet av egna noder som kan koppla till nätverket.

8.2.1 Önskade funktioner

Kryptering av signaler

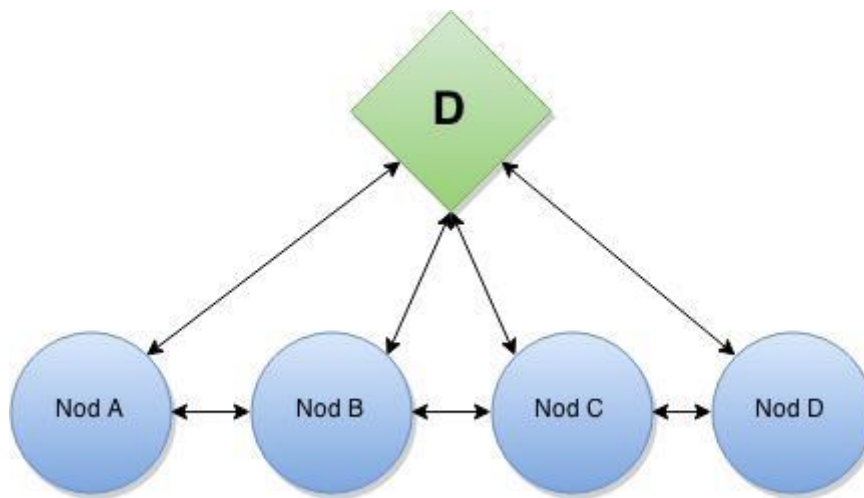
Kryptering av signalerna skulle innebära att icke-autentiserade noder inte kan ansluta till nätverket eftersom de inte kan läsa meddelandena. Detta är det mest effektiva sättet att införa säkerhet i AGA-nätverket.

Certifikat

Varje nod ska ha ett certifikat med vilket den autentiserar sig mot en proxy-nod, D i figur 11 nedan. Proxy-noden ger ut nycklar som låter autentiserade noder dekryptera/kryptera trafik. Certifikaten ska implementeras av en OEM som förmodligen även kommer agera Certificate Authority.

8.2.2 Förslag på topologier

Krypteringslösning med dagens arkitektur



Figur 11: Meshnätverk

Figur 11 ger en enkel bild över hur ett mesh-nätverk kan kommunicera. Krypteringslösningen måste anpassas till att AGA har ett mesh-nätverk. Eftersom alla användarapplikationers tuber direkt ansluts till proxy-noden så kan vi i detta fall använda den för att evaluera anslutande noder och bidrar med symmetriska sessionsnycklar som behövs för att dekryptera/kryptera trafik som skickas på nätverket. För varje signal x skulle det finnas en nyckel K_x för att dekryptera just den signalen. En signal och dess nyckel benämns som en topic och är en kryptodomän som omfattar ett flertal noder där endast de med korrekta certifikat kan läsa och skriva till topics i nätverket.

Då sessionskryptot är symmetriskt finns inget sätt att implementera olika rättigheter för att sända och ta emot signaler, då alla som har tillgång till nyckeln för att lyssna även kan skicka paket med samma krypto. Således finns inget sätt att veta vem det är som skickar eller tar emot ett meddelande vilket gör att vi inte kan bestämma olika rättigheter för att skicka och lyssna på signaler. Detta gör även en IDS mindre träffsäkert då den endast kan ge en varning om udda trafik på nätverket inte var det kommer ifrån.

Sedan finns det ingen möjlighet att kryptera data ID:t i SDP eftersom det är ett uppdelat nätverk där inte alla noder direkt ansluter till Proxy:n men är anslutna till varandra så måste

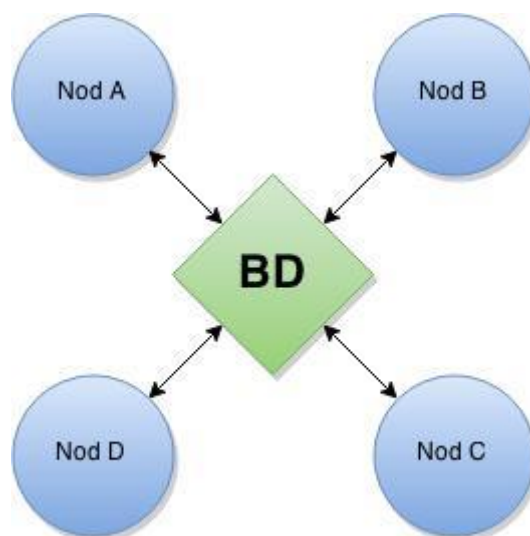
enskilda noder veta var de ska skicka paket med olika dataID:n. Alltså måste dataID:t vara okrypterat vilket ger en större attackyta för replayattacker då viktiga signaler nu lätt kan urskiljas. Men om man inför en strömkryptering skulle det inte gå att utföra en replayattack då sekvensnumret skulle vara felaktigt. Detta är dock svårt att uppnå på ett säkert sätt i ett one-to-many sammanhang som ett mesh-nätverk. Ett annat sätt är att inkludera en tidsstämpel och kasta gamla paket, detta skulle vara lättare för mesh-nätverket att implementera. Om detta implementeras krävs synkroniserade klockor då alla sändare och mottagare måste ha samma tid.

Möjligen det största problemet med dagens arkitektur är att en nod kan vara separerad från proxy-noden. Om noden är ansluten genom en annan nod kan den endast skicka sitt certifikat genom denna andra nod. Det gör att Proxy:n inte säkert kan svara med sin publika nyckel då det finns en chans att den mellanliggande noden utför en man-in-the-middle attack genom att skicka sin egna publika nyckel istället för Proxy:n:s.

Topologin är inget problem i dagens småskaliga nätverk av noder som endast existerar i en punkt, surfplatta eller liknande där de är kopplade till Proxy direkt. Men om man utökar nätverket på en större skala så finns en inbyggd flexibilitet i AGA som tillåter alla noder att koppla upp sig mot varandra och inte nödvändigtvis mot en Proxy/Directorynod vilket leder till ett mesh eller peer-to-peer-nätverk som är komplext.

Förslag på krypteringslösning med ändrad arkitektur

Det arbete som vi utfört pekar på att om AGA istället skulle kräva att alla kopplar in sig till Proxy så skulle vi ha en Broker/Directorynod, BD, som både tar hand om meddelanden och tar hand om autentisering. I ett distribuerat nätverk skulle denna BD även ha kontakt med andra proxynoder som fungerar som BD i sina respektive subnätverk.



Figur 12: Stjärnnätverk

Figur 12 presenterar arkitekturen av ett stjärn nätverk. I detta fall kan proxy-noden veta vilken nod det är som skickar meddelanden eftersom den alltid har direktkontakt med respektive nod och använder respektive kryptonyckel. Att ändra arkitekturen från ett mesh-nätverk till ett stjärn nätverk tar bort möjligheten för vanliga noder att direkt kommunicera med varandra. Proxy:n får sedan i sin tur kommunicera med andra proxy-noder om multipla AGA-nätverk krävs. Dagens arkitektur ser redan ut på detta sätt när man ansluter *applikationer* till ett litet nätverk då alla först kopplar upp sig mot proxy-noden (se Figur 3) men i ett eventuellt större nätverk skulle just nu det kunna finnas noder som kopplar direkt till varandra. Om arkitekturen ändrades så skulle kryptodomänen topics (se 8.2.2 §1 ovan) inte längre behövas utan skulle bytas ut av kommunikationskanaler mellan varje nod och proxyn.

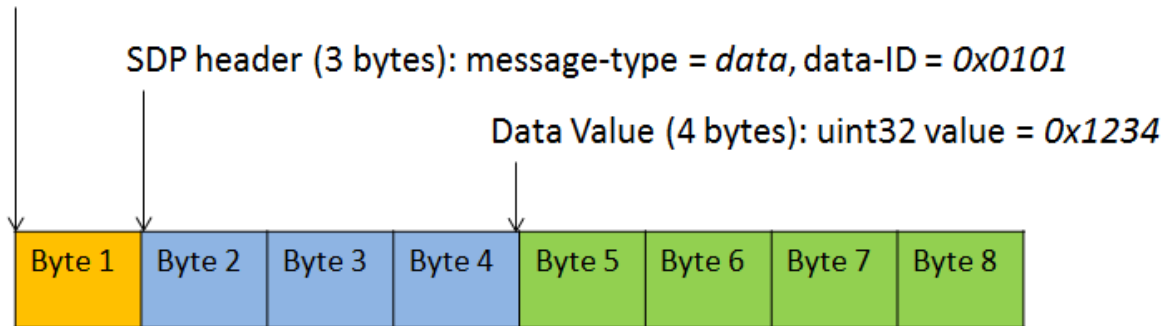
Om ändringen genomförs så kommer proxy-noden att fungera som en Message Broker och veta varifrån meddelanden kommer och kunna införa finkornig accesskontroll där en nod till exempel kan mottaga men inte skicka ett meddelande, eller till och med att noden får skicka vissa värden men inte andra. En sådan accesskontroll åtgärdar problemet med falska signaler i SDP då proxy-noden nu kan säkerställa varifrån meddelanden kommer. Att hindra falska signaler går annars bara att åstadkomma med signering vilket är väldigt resurskrävande. En ganska simpel IDS skulle vara inbyggd i proxy-noden för att upptäcka ovanliga beteenden i noder och kunna ha möjligheten att varna föraren av fordonet vid oväntad trafik.

Eftersom proxy-noden har direktkontakt med alla andra noder behövs inte en symmetrisk nyckel per topic, istället skulle varje nod ha en symmetrisk nyckel var med proxy-noden som sedan skulle dekryptera och omkryptera meddelandet till alla mottagare. Detta gör att även dataID:t på meddelanden kan vara krypterat. Proxy till nod koppling skulle även kunna vara ett strömkrypto vilket skulle vara säkrare och även skydda emot replayattacker, då det använder sessionsnummer på ett säkert sätt, strömkrypto fungerar säkrast med kryptodomäner som innehåller två noder, proxyn och en valfri nod, då ett privat sessionsnummer går att införa. Direktkontakten skulle också tillåta proxy-noden att evaluera certifikat från inkopplande noder utan risk för man-in-the-middle-attacker.

8.2.3 CAN och kryptering

CAN-bussen lämpar sig dåligt för krypterad trafik. I AGA är det troligt att AES-128 vilket ger alla krypterade paket en längd på 128. SDP är utlovad 32 bitar data per CAN-frame över ISO-TP[8]. AES-128 tar då 4 frames per meddelande. Men eftersom ISO-TP även har flow-control meddelanden[44] vilka skickas mellan frames kan ett AES-128 meddelande då ta maximalt med 8 frames och tar upp en större del av CAN-bussens bandbredd. Figur 13 visar hur ett SDP-paket på CAN-bussen ser ut.

ISO-TP header (1 byte): Single Frame (SF), data-length= 7



Figur 13: Bild över datamängd skickad i CAN[45] med SDP.

Om man studerar paketformaten i ISO-TP vilket kan ses i Tabell 6 så ser man att det går att skicka ett long CAN meddelande med consecutive frames vilket ger en mindre overhead. Detta ger att man får ett första paket med två bytes av ISO-TP information, där bit 0-3 ger signal-ID 1 för first frame och resten anger storleken.

Tabell 6: ISO-TP consecutive header [44]

First	1	size (8..4095)		Data A	Data B
Consecutive	2	index (0..15)	Data A	Data B	Data C

Detta ger att vi kan skicka 136-bitar med data i 3 frames, alltså om vi har ett 128-bitars genererat AES block så går det att skicka med endast 8 bitar med slösat datautrymme. Figur 14 visar hur mycket data som maximalt kan skickas över CAN-bussen.

1st Frame							
0	1	2	3	4	5	6	7
ISO-TP	ISO-TP	SDP	SDP	SDP	DATA	DATA	DATA

2nd Frame							
0	1	2	3	4	5	6	7
ISO-TP	DATA	DATA	DATA	DATA	DATA	DATA	DATA

3rd Frame							
0	1	2	3	4	5	6	7
ISO-TP	DATA	DATA	DATA	DATA	DATA	DATA	DATA

Figur 14: Illustration av 3 frames med data skickat med SDP över ISO-TP

9 Diskussion

Denna studie har visat på ett antal säkerhetshål i AGA som kan utnyttjas av angripare för att komma åt AGA-systemet. Flertalet existerande gränssnitt har undersökts genom intervjuer och fallstudier. Dessutom har information om redan existerande säkerhetsbrister med gränssnitten samlats. Ett antal lösningar, både på högre och lägre nivå, har därför tagits fram. Vi har hittat ett fåtal säkerhetsmekanismer som skyddar mot att applikationer ska få skicka otillåtna signaler men de var inte applicerade väl nog och kunde kringgå. Detta beror antagligen på att utvecklarna ännu inte haft tid att implementera dessa korrekt. Flera av de förslag som har tagits fram använder sig av eller förstärker dessa befintliga säkerhetsmekanismer.

Att ta hand om soft rebooten visade sig lättare än vi hade trott då det endast handlade om ett mindre programmeringsfel som fick stora effekter. Det visar på att AGA behöver en kodgranskning innan den kan anses vara säker.

En virtualiseringsmiljö har fördelen med att kunna appliceras på flera olika hårdvaror utan någon anpassning i AGA då den använder de virtuella gränssnitt som finns i hypervisorn. Vi har inte haft tid eller expertis till att ta fram en hårdvarulösning som kan filtrera SDP-trafik men förespråkar denna som en säkrare metod, detta skulle skydda fordonet mot ett infekterat AGA.

Att kryptera trafik på AGA-nätverket visade sig svårare än vi trott på grund av att vi endast hade arbetat på en surfplatta och inte ett fullskaligt system, så hade AGA fått ett naturligt stjärn nät som lämpar sig väldigt väl för kryptering. Även om detta är det sättet AGA troligen kommer att användas i fordon inom den snara framtiden så finns det funktioner som tillåter nätet att sprida sig till andra punkter och därför valde vi att diskutera ett meshnätverk i krypteringen. Efter att ha arbetat med det så framstod klara fördelar i stjärn nätverket och vi valde att eftersom AGA i nuläget har ett dedikerat team som är väl insatta och även att AGA är så pass ungt att det inte har använts i fordon så finns det fortfarande en möjlighet till att ändra i arkitekturen. Som avslutning så kommer de största problemen inte från AGA i sig självt utan att Android i sig är ett opålitligt operativsystem och med en livslängd på en bil på 17 år [46] och en livslängd på elektronik som är avsevärt kortare ställer frågan om uppdateringscykler och hur länge en bil med AGA kan hållas uppdaterad med den säkraste och senaste versionen av Android. Det är bland annat inte troligt att en dator från 1998 skulle kunna använda Windows 10. Därför är det viktigt att inte bara säkra AGA utan även se till att den opererar med tillräcklig isolering för att inte ställa till med problem för sitt fordon.

AGA kan ur miljösynpunkt användas i ett fleet-managementsystem där det kan bidra till att bättre kunna planera logistik med större transportfordon. Detta skulle leda till mindre bränsleförbrukning, och därför utsläpp, per last. Extra viktigt när det gäller stora fordon, och företag är just säkerheten. Om ett sådant system används är det viktigt att det inte ska gå att

avbryta eller spionera på. Ännu värre konsekvenser skulle kunna leda från att ett transportfordon blir 'hackat' genom AGA och leder till olyckor.

Referenser

- [1]Automotive Grade Android (2014). *Architecture*. [Wiki Article]Available at: <https://developer.lindholmen.se/redmine/projects/aga/wiki/Architecture>. [Accessed 28 May 2015].
- [2]Valasek, C. and Miller, C. (2013). *Adventures in automotive networks and controll units*.
- [3]Such, P. and Gaultier, F. (2014). *DEF CON 22 - Paul Such 0x222 and Agix - Playing with Car Firmware or How to Brick your Car*. [video] Available at: <https://www.youtube.com/watch?v=tVLatTmNI38> [Accessed 20 Apr. 2015].
- [4]Valasek, C. and Miller, C. (2014). *A Survey of Remote Automotive Attack Surfaces*.
- [5]Wind River Systems, Inc. (2015). *COMMON ANDROID SECURITY VULNERABILITIES IN AN AUTOMOTIVE ENVIRONMENT*.
- [6]NIST. (2015) *Search for Android in NVD database*. [Search Results]Available at: https://web.nvd.nist.gov/view/vuln/search-results?query=Android&search_type=last3years&cves=on
- [7]Patel, R and Davidson, B. (2011). *Forskningsmetodikens grunder – att planera, genomföra och rapportera en undersökning*. ISBN: 9789144022888
- [8]Automotive Grade Android (2014). *System Data Protocol*. [Wiki Article]Available at: <https://developer.lindholmen.se/redmine/projects/aga/wiki/SDP>. [Accessed 28 May 2015].
- [9] Mills, A and Durepos, Wiebe, E(2010) *Encyclopedia of Case Study Research*. Sage Publications. California. p. xxxi. [ISBN 978-1-4129-5670-3](https://doi.org/10.1080/10439862.2010.500000).
- [10] JetBrains, (n.d.). *IntelliJ IDEA — The Most Intelligent Java IDE*. [online] Jetbrains.com. Available at: <https://www.jetbrains.com/idea> [Accessed 11 Jun. 2015].
- [11] Robert McMillan, 2005, 'Car Whisperer Puts Hackers in the Driver's Seat', Available from: <http://www.pcworld.com/article/122077/article.html>>. [25 Apr, 2015]
- [12] Web.nvd.nist.gov, (2015). *NVD - Detail*. [online] Available at: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-4276> [Accessed 27 Apr. 2015].
- [13] Onstar.com, (n.d.). *OnStar with 4G LTE / Wi-Fi Hotspot / OnStar.com*. [online] Available at: <https://www.onstar.com/us/en/4glte/> [Accessed 20 May 2015].
- [14] Talbot, D. (2012). *4G Wireless Networks Are Extremely Vulnerable to Jamming, According to Researchers at Virginia Tech / MIT Technology Review*. [online] MIT Technology Review. Available at: <http://www.technologyreview.com/news/507381/one-simple-trick-could-disable-a-citys-4g-phone-network/> [Accessed 20 May 2015].
- [15] Srlabs.de, (2014). *Turning USB peripherals into BadUSB / Security Research Labs*. [online] Available at: <https://srlabs.de/badusb/> [Accessed 28 Apr. 2015].

- [16] Techopedia.com, (n.d.). *What is an Auxiliary Port (AUX)?* [online] Available at: <http://www.techopedia.com/definition/1356/auxiliary-port-aux> [Accessed 29 Apr. 2015].
- [17] AVOXI, (2013). *Cisco IP Phone Vulnerability / AVOXI*. [online] Available at: <http://www.avoxi.com/blog/cisco-ip-phone-vulnerability/> [Accessed 29 Apr. 2015].
- [18] autosec, (2015). *Indirect physical channels*. [online] Available at: <http://www.autosec.org/pubs/cars-usenixsec2011.pdf> [Accessed 29 Apr. 2015].
- [19] Newcomb, D. (2014). *Tapping Into In-Car WiFi | Edmunds.com*. [online] Edmunds. Available at: <http://www.edmunds.com/car-technology/tapping-into-in-car-wifi.html> [Accessed 30 Apr. 2015]
- [20] Rudhrakumar Venkatesa, Shashidhar Lakkavalli. (n.d.). *TCP/IP Vulnerabilities*. [online] Available at: <http://cs.ucsb.edu/~koc/ns/projects/00Reports/LV.pdf> [Accessed 21 May 2015].
- [21] Xsechosting.co.uk, (n.d.). [online] Available at: http://www.xsechosting.co.uk/cms/index2.php?option=com_content&do_pdf=1&id=17 [Accessed 9 Jun. 2015].
- [22] Erland Jonsson, Magnus Almgren. (n.d.). *Network Attacks, part 2*. [online] Available at: <http://www.cse.chalmers.se/edu/course/EDA263/oh15/L07%20Network%20Attacks,%20part%202--print.pdf> [Accessed 21 May 2015].
- [23] Allaboutcookies.org, (n.d.). *What is a Web Browser?*. [online] Available at: <http://www.allaboutcookies.org/browsers/> [Accessed 11 May 2015].
- [24] Spiceworks, I. (2014). *Android.Fitikser is a Trojan horse for Android devices that steals information*. [online] The Spiceworks Community. Available at: <http://community.spiceworks.com/topic/597154-Android-fitikser-is-a-trojan-horse-for-Android-devices-that-steals-information> [Accessed 11 Jun. 2015].
- [25] Williams, M. (2015). *BMW cars found vulnerable in Connected Drive hack*. [online] PCWorld. Available at: <http://www.pcworld.com/article/2878437/bmw-cars-found-vulnerable-in-connected-drive-hack.html> [Accessed 27 Apr. 2015].
- [26] About-threats.trendmicro.com. (2015) *12 Most Abused Android App Permissions*. [online]. Available from: <http://about-threats.trendmicro.com/us/library/image-gallery/12-most-abused-Android-app-permissions> [Accessed 28 May 2015].
- [27] Spence, E. (2015). *New Android Malware Strikes At Millions Of Smartphones*. Forbes [online]. Available from: <http://www.forbes.com/sites/ewanspence/2015/02/04/Android-malware-apps-deleted/> [Accessed 28 May 2015].
- [28] Xen Project. (2015). *What is the Xen Project Hypervisor*. [Wiki Article]. Available from: http://wiki.Xenproject.org/wiki/Xen_Overview#What_is_the_Xen_Project_Hypervisor.3F [Accessed 28 May 2015]
- [29] Itsecurity.telelink.com, (n.d.). *IT Security – Hyperjacking*. [online] Available at: <http://itsecurity.telelink.com/hyperjacking/> [Accessed 12 May 2015].

- [30] J. Schwartz, M. (2012). *New Virtualization Vulnerability Allows Escape To Hypervisor Attacks*. [online] Dark Reading. Available at: <http://www.darkreading.com/risk-management/new-virtualization-vulnerability-allows-escape-to-hypervisor-attacks/d/d-id/1104823?> [Accessed 12 May 2015].
- [31] Support.avg.com, (n.d.). *What is a rootkit / AVG*. [online] Available at: https://support.avg.com/SupportArticleView?l=en_US&urlname=What-is-rootkit [Accessed 12 May 2015].
- [32] Rutkowska, J. (2006). *The Invisible Things Lab's blog: Introducing Blue Pill*. [online] Theinvisiblethings.blogspot.se. Available at: <http://theinvisiblethings.blogspot.se/2006/06/introducing-blue-pill.html> [Accessed 12 May 2015].
- [33] [TPM] Trusted Computing Group (n.d.) *TPM Main Specification*. Available from: http://www.trustedcomputinggroup.org/resources/tpm_main_specification [Accessed 28 May 2015].
- [34] [Public key infrastructure] Net Security Training, 2015, *What is Public Key Infrastructure*. 6th Floor, Craven house, 40 Uxbridge Road, London. Available from: <http://www.net-security-training.co.uk/what-is-a-public-key-infrastructure/> [Accessed 28 May 2015].
- [35] Scarfone, K and Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS) Computer Security Resource Center (National Institute of Standards and Technology) (800–94)*.
- [36] Janssen, C. (n.d.). *What is a Hypervisor? - Definition from Techopedia*. [online] Techopedia.com. Available at: <http://www.techopedia.com/definition/4790/hypervisor> [Accessed 28 May 2015].
- [37] Collabprojects.Linuxfoundation.org. (2015). *Linux Foundation Collaborative Projects*. [online]. 2015. [Accessed 28 May 2015]. Available from: <http://collabprojects.Linuxfoundation.org/>
- [38] Wiki.xen.org, (n.d.). *Xen Project Software Overview - Xen*. [online] Available at: http://wiki.xen.org/wiki/Xen_Project_Software_Overview [Accessed 13 May 2015].
- [39] Haughn, M. and J. Bigelow, S. (2015). *What is ARM processor? - Definition from WhatIs.com*. [online] WhatIs.com. Available at: <http://whatis.techtarget.com/definition/ARM-processor> [Accessed 18 May 2015].
- [40] Genode.org, (n.d.). *Genode - An Exploration of ARM TrustZone Technology*. [online] Available at: <http://genode.org/documentation/articles/trustzone> [Accessed 18 May 2015].
- [41] Rosenberg, D. (2014). [online] Available at: <https://www.bl.a.ckhat.com/docs/us-14/materials/us-14-Rosenberg-Reflections-On-Trusting-TrustZone-WP.pdf> [Accessed 28 May 2015].

- [42] RAVINDRAN, K. and XU, X. (2015). *AUTOSAR and Linux – Single chip solution*. [online] Chalmers. Available at: <http://publications.lib.chalmers.se/records/fulltext/214009/214009.pdf> [Accessed 28 May 2015].
- [43] Cvedetails.com, (2015). *Xen : Security vulnerabilities*. [online] Available at: http://www.cvedetails.com/vulnerability-list/vendor_id-6276/Xen.html [Accessed 12 May 2015]
- [44] ISO.ORG. (2011). Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. [Accessed 28 May 2015] Available from: <https://www.iso.org/obp/ui/#iso:std:iso:15765:-2:ed-2:v1:en>
- [45] Automotive Grade Android (2014). *SDP 2.0*. [Powerpoint Presentation] Available at: <https://developer.lindholmen.se/redmine/projects/aga/wiki/SDP>. [Accessed 28 May 2015].
- [46] Mazda, (n.d.). *Fordon som nått slutet av sin livslängd*. [online] Välkommen till Mazda Sverige - Upplev vårt spännande modellprogram. Available at: <http://www.mazda.se/om-mazda/miljo/atervinning/> [Accessed 11 Jun. 2015].
- [47] Lindmark, A. (2015) *An App Approval Process for Commercial Vehicles*. Göteborg : Chalmers University of Technology
- [48] Bosch, R. (1995). *Can specification*. [online] Available at: <http://www.kvaser.com/software/7330130980914/V1/can2spec.pdf> [Accessed 25 Jun. 2015].
- [49] Whatismyipaddress.com, (n.d.). *What is a Gateway?*. [online] Available at: <http://whatismyipaddress.com/gateway> [Accessed 25 Jun. 2015].
- [50] Fischer, S. (2008) *Analysis of Lightweight Stream Ciphers* [online] Available at: http://biblion.epfl.ch/EPFL/theses/2008/4040/EPFL_TH4040.pdf [Accessed 30 July 2015]

Bilaga

BILAGA A. Sid 1(1)

```
protected void sendGreeting(final SocketChannel socketChannel) {  
  
    final byte[] greeting = Constants.GREETING.getBytes();  
    final byte[] message = new byte[greeting.length + 2];  
  
    System.arraycopy(greeting, 0, message, 0, greeting.length);  
    message[greeting.length] = 0;  
    message[greeting.length + 1] = 0;  
  
    try {  
  
        final ByteBuffer byteBuffer = ByteBuffer.wrap(message);  
  
        synchronized (socketChannel) {  
  
            while (byteBuffer.hasRemaining()) {  
  
                socketChannel.write(byteBuffer);    //Här sker ett exception som inte hanteras  
  
            }  
  
        }  
  
    } catch (ClosedChannelException e) {  
  
        LOGGER.info("Channel was closed from another thread: " + e.getMessage());  
  
    } catch (IOException e) {  
  
        LOGGER.error(e.getMessage());  
  
        throw new SDPRuntimeException(e);  
  
    }  
  
}
```