Automotive Control Systems Security

Where am I going

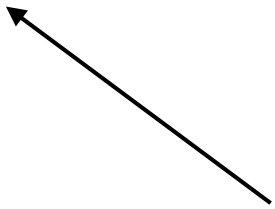
and

Why am I in this handbasket?

Peter Gutmann

University of Auckland

1

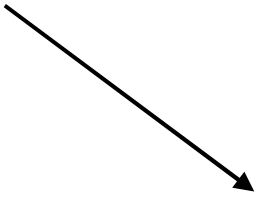


"format your slides for 16:9 aspect ratio"

$4^2 = 16, 3^2 = 9$

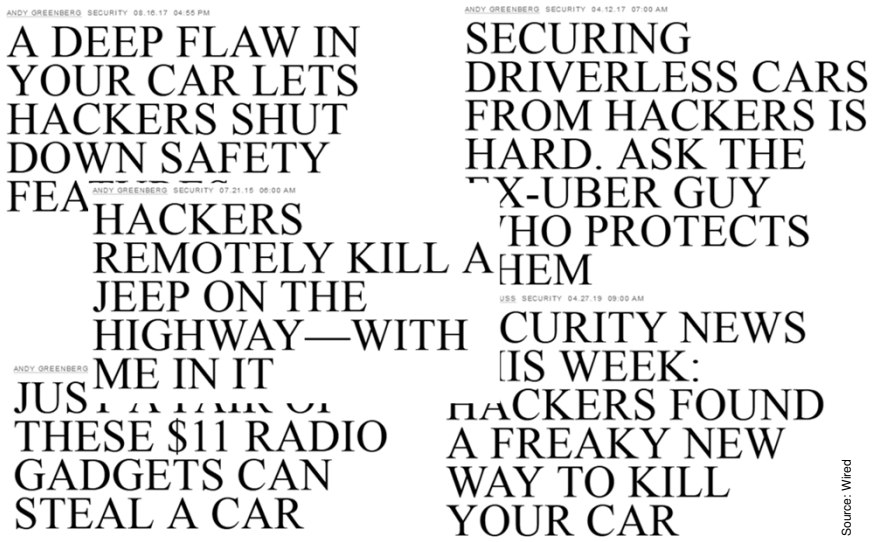
$\Rightarrow$ 4:3 $\equiv$ 16:9

$\Rightarrow$ This is 16:9. QED



2

Automotive (In)security

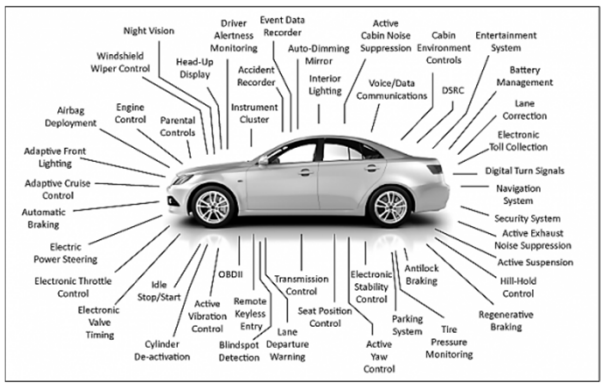


Source: Wired

Automotive (In)security (ctd)

Source: Wired

Automotive Control Systems



Source: eeNews Automotive

A vehicle in 2018 typically contains 100-300 microcontrollers or processors, 50+ complex electronic control units, between 5 and 20 million lines of software code

— eeNews Automotive

Automotive Control Systems (ctd)

Standard access path for attackers: Bluetooth (or similar) to the head unit

Hands-free phoning, music streaming, navigation, …

Automotive Control Systems (ctd)

Head unit runs BT radio at high power

- Don't want any interruptions to the music streaming
- Also means your head unit is remotely accessible from five cars away

Source: RCDriver

Automotive Control Systems (ctd)

Source: RCDriver

Automotive Control Systems (ctd)

But wait, there's more! Remember this icon?

MirrorLink turns the promise of the connected car into reality
— Car Connectivity Consortium

IP, USB, Bluetooth, WiFi, VNC, RTP, UPnP

- telnet, carrier pigeons, smoke signals…

Some of these aren't even protocols, they're security holes with wire formats

Automotive Control Systems (ctd)

And it gets even worse…

- Increasing moves to allow vehicle control via cellphone apps
- Either manufacturer-installed remote access to the head unit or aftermarket add-ons

Source: AutoBlog

Occasionally: OBD-II dongle with Bluetooth interface

- Typically Chinese clones of ELM327 with buggy firmware

Automotive Control Systems (ctd)

And the underlying OS

- QNX
- Windows CE
- Android
- Custom RTOS

Windows CE running UPnP

QNX running VNC

Android running RTP

- What could possibly go wrong?

Source: Xtrons

At least one of these is right

Automotive Control Systems (ctd)

So that's the head unit

The rest is very different…

AUTOSAR

Automotive Open System Architecture

- Sorry, AUTomotive Open System ARchitecture

Founded by BMW, Bosch, Continental, Daimler Chrysler, Siemens, and VW

- Based on an earlier standard OSEK / ISO 17356 from much the same players
- Later joined by others, Ford, GM, Honda, Hyundai, Nissan, Peugot, Renault, Tata, Toyota, Volvo

Define a standard software architecture for ECUs and related systems

- And, eventually, much, much more

AUTOSAR Goals

Created standards for both dependability and security

- Not so much an API as an architecture for automotive safety instrumented systems

There are various others, but AUTOSAR is

- Comprehensive
- Involves a large number of manufacturers
- Covers both standard ECU goals and security

Understanding AUTOSAR or an equivalent is necessary to help understand automotive security issues

AUTOSAR Goals (ctd)

Primary goal: Dependability

> The trustworthiness of a system such that reliance can justifiably be placed on the service it delivers – the delivered service being the system's behaviour as perceived by the user
>
> — "Dependability: A Unifying Concept for Reliable, Safe, Secure Computing"

- Shared with other safety instrumented/critical control systems, e.g. avionics

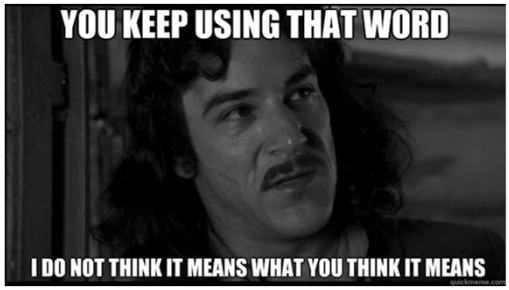
Need to understand this goal in order to understand how it interacts with security

AUTOSAR Goals (ctd)

The term "trustworthy" (and "trust") in the context of dependability is nothing like the "trust" of computer security

- "The CA is a trusted third party"
- "The Intel Management Engine is a trusted component"
- ARM TrustZone

In all of these cases "trust"really means "is forced to rely upon"



Dependability

Dependable systems can experience faults

- A fault doesn't necessarily reduce the dependability of a system

Fault must result in an error in order to cause a problem

If the error propagates beyond a system barrier so that it becomes visible to the rest of the system, it becomes a failure

> A fault can manifest itself as an error […] and the error can ultimately cause a failure
>
> — ISO 26262, "Road Vehicles — Functional Safety"

Dependability (ctd)

Only the full progression fault → error → failure is a visible problem

- Handled via fault detection, isolation, and recovery (FDIR)

Example from computer networking

- Fault: Electrical glitch induced onto Ethernet cable
- Error: Corrupted data packet
  - Detection: Failed CRC/FCS check
  - Isolation: Packet is dropped
  - Recovery: Kicked upstairs, typically TCP-level
- Failure: None, fault mitigated

Dependability (ctd)

Fault mitigation in automotive systems

- Is the value within a range of plausible values?
  - Engine temperature, vehicle speed, etc
  - Unless the vehicle is powered by a Mr.Fusion, an engine temperature of 3000°C is suspect
- Is the combination of values within a range of plausible values?
  - Engine speed / vehicle speed / gear ratio
- Do multiple redundant sources agree?
  - Angle-of-attack sensors on aircraft
- Exotic rigorous solutions
  - Predictor/corrector models like Kalman filters

Dependability (ctd)

Signal metrics

- Signal quality, timestamps, sequence numbers, signal-changed status

Timing protection

- Protecting from activities that take too long to complete
- Excessive runtime upsets response-time guarantees for other components

Mitigations

Substitute values

- If a value is implausible, substitute an approximation to use in subsequent calculations
- Malfunctioning sensor, use last known good value

Voting / redundancy

- 2oo3 or similar mechanisms

Liveness monitoring of subsystems

- Watchdogs, heartbeats

Diverse monitoring

- External monitor ensures the system remains within safety bounds

Mitigations (ctd)

Execution sequence monitoring

- Check control flow graph (CFG)
- Monitor control flow through basic blocks
  - As a convenient side-effect, severely hampers ROP

Reliability trumps everything

- "Limp home" mode as a design safe state
- Disable some subsystems, e.g. keep ABS (anti-lock braking) but no ESC (electronic stability control)
- c.f. MEL in aircraft
  - Minimum (functioning) equipment list for an aircraft to be considered airworthy

Fault-Tolerance

Not just a fancy name, the system is literally tolerant of faults

- A great deal of engineering effort goes into providing this capability

Overreacting to faults can actually be harmful

> In some situations taking recovery actions due to errors […] may cause more damage than it does good. Reacting to such errors may cause an over-reaction where the recovery actions may put the system in a state where it is less safe than previously
>
> — "Explanation of Error Handling on Application Level", AUTOSAR

Fault-tolerance is the diametric opposite of what crypto/ security does

Fault-Intolerance

In crypto/security, the goal is to find the single bit that's out of place

- One single bit out of place → fail

```
a. If the length of L is greater than the input
   limitation for the hash function (2^61 - 1 octets
   for SHA-1), output "decryption error" and stop.

b. If the length of the ciphertext C is not k octets,
   output "decryption error" and stop.

c. If k < 2hLen + 2, output "decryption error" and
   stop.
```

— PKCS #1 v2.1

- "… and stop" means "fault and error and failure" all in one

Fault-Intolerance (ctd)

Once you've found the discrepancy, you've won

Source: LA Times

No known standard covers how to continue after this point

- c.f. vast literature on fault tolerance and error recovery

Fault Mitigation vs. Security

Plausibility checks: Binary yes/no

Execution sequence monitoring: No

Substitute values: No

Voting/redundancy: No (except in Type 1 crypto hardware)

Liveness checks/signal metrics: N/A

Timing protection: Public-key crypto operations are variable-time

- Some operations like keygen only terminate probabilistically

Fault Mitigation vs. Security (ctd)

Continuing with degraded functionality

Fault Mitigation vs. Security (ctd)

Once you've found the discrepancy, you've won

- Why would you want to continue?

This makes automotive security… interesting

- Irresistible force, meet immovable object

See "Hard and Not-necessarily-hard Problems in Cryptography", `https://www.cs.auckland.ac.nz/~pgut001/pubs/problems.pdf`

- Specifically, coverage of wicked problems and availability issues

The AUTOSAR Environment

OSEK/VDK, rebranded AUTOSAR Classic, is a fairly standard high-assurance RTOS

System configuration is defined statically at build time

> If tasks are created dynamically then things are much more complex; timing predictions may be filed under 'fiction'
>
> — "Real-Time Operating Systems", Vol.1

- Makes reasoning about issues like deadlocks much easier

No dynamic allocation of any kind

The AUTOSAR Environment (ctd)

All tasks execute out of non-volatile memory

- No code in RAM
- Also makes IPL very quick
  - '96 Toyota ECU can be restarted at 180 kph
  - Only noticeable effect was a brief ignition ping
  - Had gone into limp-home mode at Hampton Downs track
  - That doesn't mean you *should* do it…
- Fault mitigation via rejuvenation

The AUTOSAR Environment (ctd)

Scheduling is typically static, e.g. rate-monotonic or heuristic scheduling

- OSEK is somewhat open-ended on this, but standard scheduling strategies from conventional OSes don't work in an RTOS

No recursion, longjmp

- From MISRA / Motor Industry Software Reliability Association, required by AUTOSAR
- Many embedded compilers have MISRA compliance checking built in
- Also supported by third-party static analysers, Coverity/ cppcheck/Goanna/PC-Lint/Polyspace/PVS-Studio/etc

The AUTOSAR Environment (ctd)

More rules from JSF-AV and SEI CERT Coding Standard

Fixed stack size

Fixed task priorities

etc

The AUTOSAR Environment (ctd)

Certification requirements are onerous

- Make safety claims for the device, e.g. "meets all the requirements of ISO 26262"
- Provide arguments to support the claim, e.g. via formal notation
- Provide evidence to support the arguments

An argument without supporting evidence is unfounded, and therefore unconvincing

— ISO 26262-10

Example tools: Fault tree analysis, discrete event simulation, Petri nets (IEC 61508-3), formal methods like Promela/SPIN (ISO 26262-6), etc

The AUTOSAR Environment (ctd)

In order to be road legal, a car must have a certificate of conformity

- US self-certifies

  U.S. automakers self-certify that they are meeting U.S. vehicle standards

  — "U.S. and EU Motor Vehicle Standards"

  – Like Boeing with the 737 MAX
- Europe has Whole Vehicle Type Approval, WVTA
- Rest of the world is covered by similar UN rules

Outside the US, once type approved, systems can't be changed unless the change gets type approval

The AUTOSAR Environment (ctd)

But Tesla does updates all the time!

- Tesla are self-certifying (OK in the US)…
- … and self-insured if there's a problem

Tesla's famous OTA brake system update would be illegal anywhere but the US

- Even in the US it was highly risky legally

Tesla Model 3 Gets CR Recommendation After Braking Update

Automaker responds to Consumer Reports test results and reduces stopping distance by nearly 20 feet

Until now, that type of remote improvement to a car's basic functionality had been unheard of. "I've been at CR for 19 years and tested more than 1,000 cars," says Jake Fisher, director of auto testing at Consumer Reports, "and I've never seen a car that could improve its track performance with an over-the-air update."

The AUTOSAR Environment (ctd)

So how does Tesla sell outside the US?

- Long story, they have WVTA on the models sold outside the US
- Other vendors don't want to rock the boat over OTA changes made to WVTA vehicles because of their own OTA plans
- As soon as there's a fatal accident, things will get interesting because Tesla's OTA will probably be found to be illegal
- Being discussed by the International Organization of Motor Vehicle Manufacturers (OICA, Organisation Internationale des Constructeurs d'Automobiles) of which Tesla isn't a member

UN Task Force on Cyber Security and (OTA) software updates (CS/OTA) has been working out how to deal with OTA

The AUTOSAR Environment (ctd)

Typical target CPUs

- 68HC08, 68HC12
- MPC5xx, MAC7x00
- RH/V850x
- TDA2x, TDA3x
- TriCore

Source: OBD2Express

The AUTOSAR Environment (ctd)

Overview

The MPC555 MCU offers a high-speed solution designed for automotive applications such as engine and transmission control, as well as robotics and avionics control.

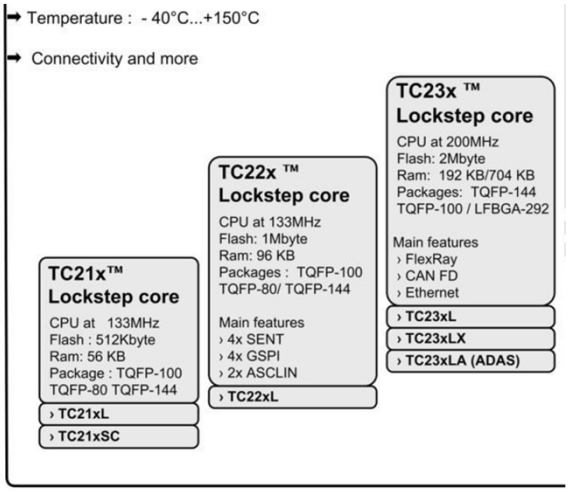
Features

- 40 MHz core with floating point unit
- 26 KB of static RAM
- 448 KB flash EEPROM memory with 5-volt programming (CMF)
- Flexible memory protection unit
- GPIO support
- Two time processor units (TPU3)
- 18-channel modular I/O system (MIOS1)
- Two queued analog-to-digital converter modules (QADC)
- Two CAN 2.0B controller modules (TouCANs)
- Queued serial multi-channel module (QSMCM)
- U-bus system interface unit (USIU)

Source: NXP

- Value is the masses of automotive capabilities, not the CPU power

The AUTOSAR Environment (ctd)



Source: Infineon

- Many safety-oriented features, e.g. lockstep cores, extensive error checking/ correction, ASIL-D safety classification
  - Caution: Marketing, ASIL inflation
- Note Milspec-equivalent temperature range

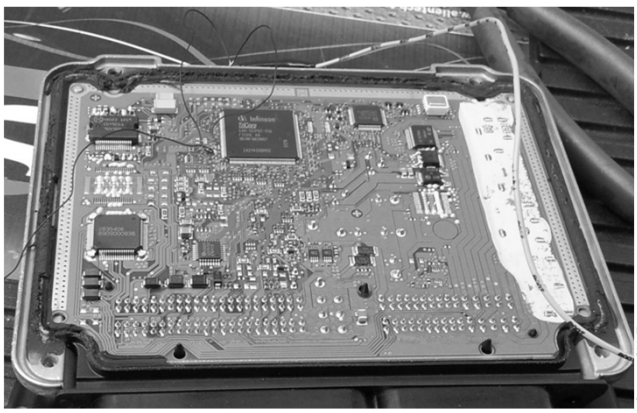
The AUTOSAR Environment (ctd)

Example: Bosch EDC17 ECU

- Used in many European cars, including VW diesels
- Yes, *those* diesels

TriCore 1797

- 180 MHz
- 128kB RAM
- 2-4MB flash (ROM)
- 64kB flash (writeable)


Source: ChipTuners

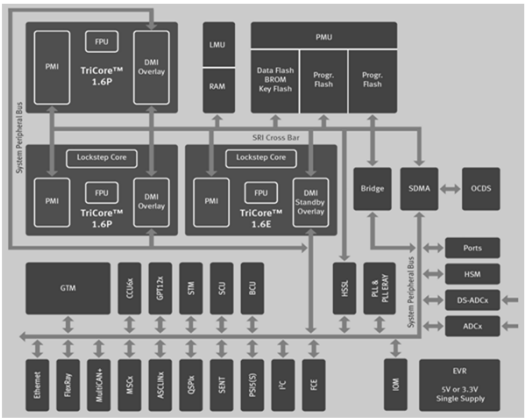
The AUTOSAR Environment (ctd)

To put that into terms people can understand…

- 0.2 GHz single-core CPU
- 0.0001 GB RAM
- 0.000002 TB hard drive

Source: HP

The AUTOSAR Environment (ctd)



Source: Infineon

Again note the value proposition, it's the I/O capabilities (alongside the safety features)

Intermezzo

Tricky to distinguish reality from marketing

- Various products covered in presentations and glossy brochures but probably not readily available
- Available real soon now, in the next revision, once you throw out your existing hardware and redesign with new devices

This talk is an attempt to capture today's reality, not future dreams

Selection criterion: What would you find in Joe Sixpack's garage?

- With a little input from ECU tuners

The AUTOSAR Environment (ctd)

The memory overhead of RTA-OSEK is:

Memory type	Overhead (bytes)
RAM	28
ROM/Flash	204

In addition to the RTOS overhead, each object used by an application has the following memory requirements:

Object	RAM Bytes	ROM Bytes
BCC1 task	0	36
BCC2 task	10	52
ECC1 task	28	60
ECC2 task	30	68
Category 1 ISR	0	0
Category 2 ISR	0	64
Resource	0	20
Internal Resource	0	0
Event	0	4
Alarm	12	60
Counter	4	44
Taskset (RW)	4	4
Taskset (RO)	0	4
Schedule	16	36
Arrivalpoint (RW)	12	12
Arrivalpoint (RO)	0	12

Source: ETAS

- BCC/ECC = basic/ extended conformance class tasks (details unimportant)
- 0 bytes RAM from statically defined task, ROM usage is the TCB

The AUTOSAR Environment (ctd)

Sample application configuration

ECC1

The ECC1 application uses 7 basic tasks and 1 extended task with unique priorities. Task H is the extended task and it waits on a single event that is set by basic tasks A-G.

This application has the following overheads:

Memory usage	Bytes
OS ROM	**2782**
OS RAM	**277**
comprising RAM data	156
comprising RAM stack	121

BCC1

The BCC1 application uses 8 basic tasks with unique priorities.

This application has the following overheads:

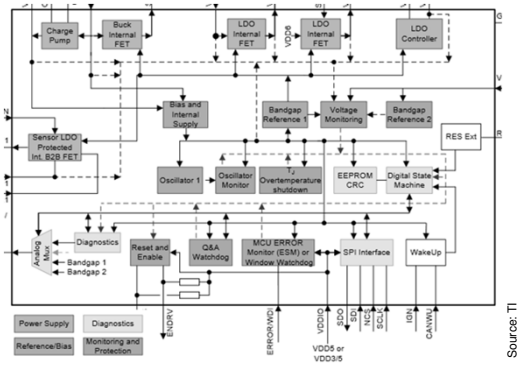
Memory usage	Bytes
OS ROM	**2032**
OS RAM	**232**
comprising RAM data	128
comprising RAM stack	104

The AUTOSAR Environment (ctd)

Moore's Law doesn't apply

Emphasis is on reliability and safety, not bleeding-edge performance

- Designed and certified for use in particularly harsh environments
- If this stuff wasn't classed as automotive, it'd be export-controlled
- Often more functionality for safety than primary



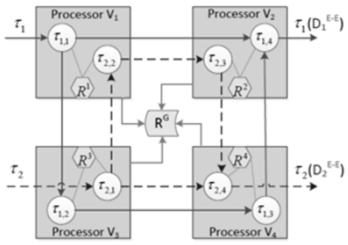
Source: TI

The AUTOSAR Environment (ctd)

Distributed real-time control system, not a single computer

100-300 microcontrollers or processors, 50+ complex electronic control units, between 5 and 20 million lines of software code
— eeNews Automotive

- Control system is the sum of the individual parts, not a single ECU somewhere
- Complex timing and communications constraints
- For perimeter-security based thinking, the entire system is inside the perimeter — all buses are internal



Source: TU Braunschweig

Intermezzo

This is OSEK/VDK, rebranded AUTOSAR Classic

- AUTOSAR NG is different

Based on the well-known formally-verified realtime OS, Linux

- Individual high-assurance ECUs become VMs on a single Linux system

```
[<ffffffff81549f6e>] ? panic+0xa7/0x179
[<ffffffff8107d98e>] ? vprintk_default+0xe/0x10
[<ffffffff81c3c541>] ? mount_block_root+0x216/0x2cb
[<ffffffff81c6beb3>] ? gunzip+0x0/0x34e
[<ffffffff81002930>] ? bstat+0x200/0x8d0
[<ffffffff81c3c64c>] ? mount_root+0x56/0x5a
[<ffffffff81c3c7c0>] ? prepare_namespace+0x170/0x1a9
[<ffffffff81c3babd>] ? kernel_init+0x2e1/0x2f7
[<ffffffff8100969d>] ? __switch_to+0x7d/0x340
[<ffffffff8100c28a>] ? child_rip+0xa/0x20
[<ffffffff81c3b7dc>] ? kernel_init+0x0/0x2f7
[<ffffffff8100c280>] ? child_rip+0x0/0x20
```

- Note to self: Buy lifetime train travel pass

The AUTOSAR Environment (ctd)

Devices have a design life of ten to twenty years

- There is hardware deployed today that was designed when the people now maintaining it were in kindergarten

Firmware is never updated, and frequently can never be updated

- Storage isn't writeable
- There's no room
- It's too risky to update due to bricking risk or decertification of the system

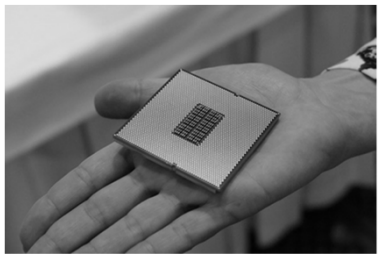
Avoiding recertification is a huge motivator

- 737 MAX is a very visible example of how motivated companies are to avoid it

The AUTOSAR Environment (ctd)

What about using *$shiny_new_thing*?

- This is very hard for geeks to resist
- "*$corporation* has just announced *$shiny_new_thing*, this will solve all of your problems"



Source: PCPerspective

The AUTOSAR Environment (ctd)

Translated: You need to redo your

- Product roadmap
- Supplier agreements
- Second-source/LTS
- Licensing
- Hardware design
- Software toolchain
- BSP
- Firmware
- Testing
- Certification

Thanks, but no thanks

The AUTOSAR Environment (ctd)

Examples

- Compile with optimisation disabled since this destroys the 1:1 mapping of source → object code
  - IEC 61508-3 §7.4.4.4 / ISO 26262-8 §11.4.4.2 warn against opt.compilers
  - See "Software security in the presence of faults" talk
- Built on 1990s-vintage PCs scrounged from eBay because that's what was certified

Source: eBay

Now that the scene is set, let's look at crypto and security…

AUTOSAR Security

What crypto resources are available?

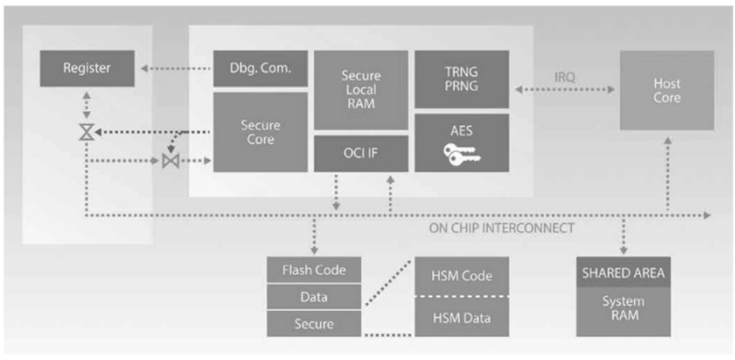
- AES, supported on higher-end SoCs
- Some form of RNG for keygen, reasonably common
  - Or can use environmental sources
- SHA-1, DES occasionally
- RSA, ECDSA is practically non-existent
- Everything else is *actually* non-existent

Any crypto solution had better be based pretty exclusively on AES

- As a convenient side-effect, won't have to worry about which PKC will be in fashion in ten years' time or what keysize they're wearing in Paris that year

AUTOSAR Security (ctd)

Crypto is implemented as IP cores added to the SoC, not inline instructions



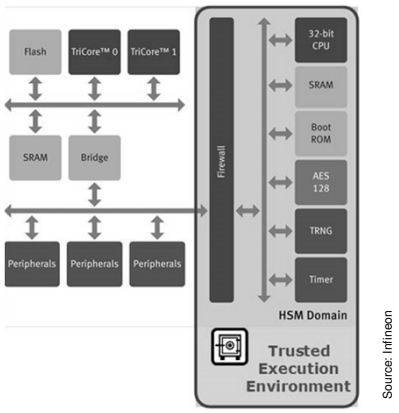
Source: ETAS

This is a hypothetical "automotive HSM" from Bosch

- Few automotive SoCs actually have any crypto hardware

AUTOSAR Security (ctd)

Some newer SoCs have crypto hardware support

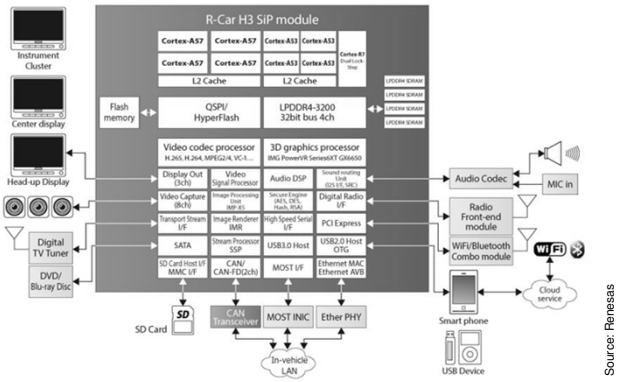


But this is few and far between

- Note the 2001-vintage crypto

AUTOSAR Security (ctd)

The only CPUs that do have crypto support are standard tablet/set-top-box style ones meant for head units



This is also the point of entry to the car for attackers

AUTOSAR Security (ctd)

Programming interfaces to the crypto IP in the SoCs are all vendor-specific

- Invariably hard to use
- Need to talk PIO or DMA
- The HAL or vendor firmware may not make it available
- Access is almost always slower than doing it natively in software

We can solve all our problems (except the speed one) with a standardised API layer!

Crypto HALs

PKCS #11

- Has been around for 25 years
- The standard interface to security hardware
- OO interface using C API

```
CK_ATTRIBUTE publicKeyTemplate[] = {
        { CKA_CLASS, CKO_PUBLIC_KEY, sizeof( CK_OBJECT_CLASS ) },
        { CKA_KEY_TYPE, CKK_RSA, sizeof( CK_KEY_TYPE ) },
        { CKA_VERIFY, CK_TRUE, sizeof( CK_BBOOL ) },
        { CKA_ENCRYPT, CK_TRUE, sizeof( CK_BBOOL ) },
        { CKA_MODULUS, modulus, modulusLength },
        { CKA_PUBLIC_EXPONENT, exponent, exponentLength },
        };

C_CreateObject( hSession, publicKeyTemplate, 6, &hRsaKey );
C_EncryptInit( hSession, CKM_RSA_PKCS, hRsaKey );
C_Encrypt( hSession, inData, inLength, outData, &outLength );
```

Crypto HALs (ctd)

Core document (for v2.01) is ~150 pages, plus a 60-page catalogue of crypto mechanisms

- Core API is a thin shim over the underlying hardware capabilities
- Most of the work is marshalling and unmarshalling

Bloated up by optional PIN management, SSO/user handling, etc

- This is all optional, can implement just the basic functionality

Crypto HALs (ctd)

This is way too…

- Complicated
- Heavyweight
- Unhip

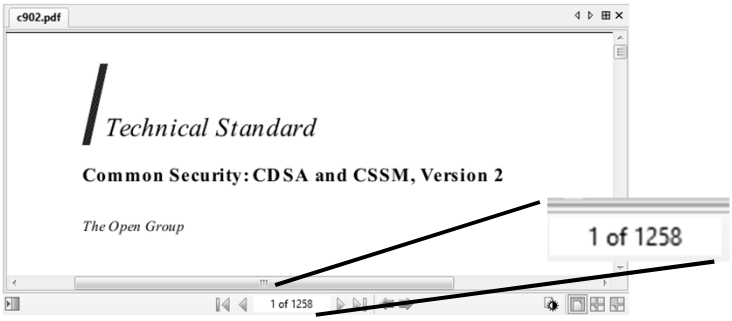
Let's design our own!

- We can do it better than those guys

Crypto HALs (ctd)

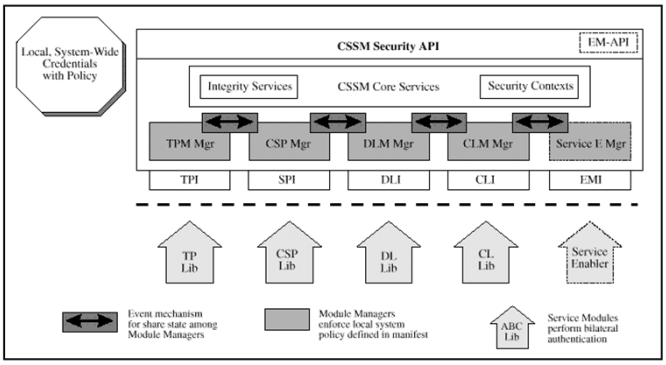
Common Data Security Architecture

- Relatively simple design and API in 1.0, from Intel Architecture Labs
- Did exactly the same thing as PKCS #11, only differently
- Then the Open Group decided to standardise it



Crypto HALs (ctd)

Collapsed under its own weight



Source: Open Group

Crypto HALs (ctd)

No-one ever managed to implement it, although Apple tried very hard

- 3-4 years work by a full-time team

CDSA was "a classic late-1990s API", "a bloated, unmanageable mess" (CDSA developer)

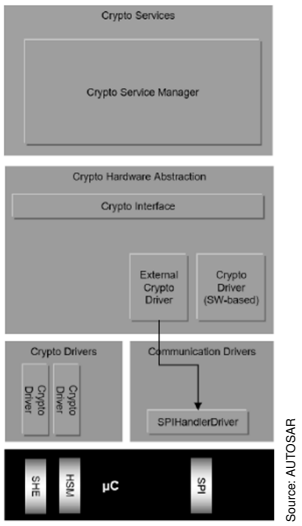
AUTOSAR Crypto API

AUTOSAR approach

- Ongoing since 2010
- 486 pages of specification (so far)
- It's OK, I've read them for you (ugh)

Attempt to reinvent CDSA

- Not explicitly, but this is what the kitchen-sink abstract-concept approach to design naturally produces



Source: AUTOSAR

AUTOSAR Crypto API (ctd)

Or maybe an attempt to reinvent PKCS #11

- Initialization with Start function (e.g. Csm_SymEncryptStart)
- Update function (e.g. Csm_SymEncryptUpdate)
- Finish function (e.g. Csm_SymEncryptFinish)

Source: AUTOSAR

Encryption functions	C_EncryptInit	initializes an encryption operation
	C_Encrypt	encrypts single-part data
	C_EncryptUpdate	continues a multiple-part encryption operation
	C_EncryptFinal	finishes a multiple-part encryption operation

Source: PKCS #11

This is a near-exact copy of PKCS #11 done twenty years later

- Looks like they reinvented all the long-since-closed PKCS #11 security holes as well…

AUTOSAR Crypto API (ctd)

AUTOSAR crypto diagrams look frighteningly similar to diagrams from the CDSA spec

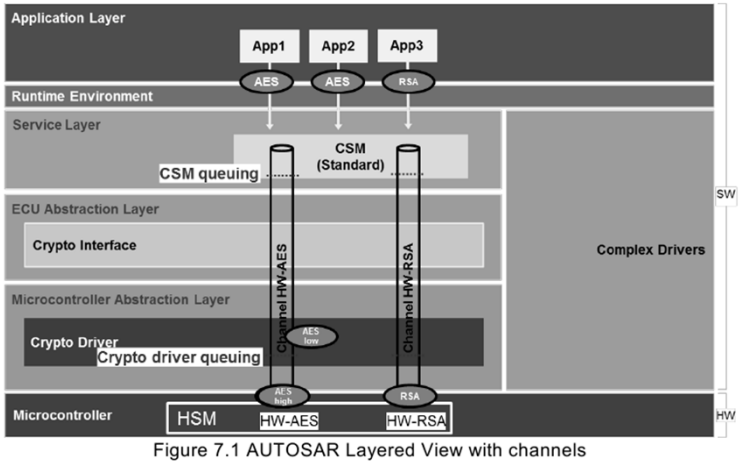


Figure 7.1 AUTOSAR Layered View with channels

Source: AUTOSAR

AUTOSAR Crypto API (ctd)

The following cryptographic algorithms or primitives should be supported by the Crypto Stack:

- Random Number Generation
  - Deterministic Random Number Generator (DRNG)
  - True Random Number Generator (TRNG)
- Symmetric Encryption
  - AES
    - Key Length: 128 and 256 bits
    - Modes: ECB, CBC, CTR, GCM, OFB, CFB, XTS
  - PRESENT
    - Key Length: 128 bits
    - Modes: ECB, CBC, CTR, GCM, OFB, CFB, XTS
  - ChaCha12/ChaCha20
    - Key Length: 256 bits
- Asymmetric Encryption/Decryption and Signature Handling
  - RSA
    - Key Length: 1024, 2048, 3072, 4096
    - Padding: PKCS#1 v2.2
  - Curve25519/Ed25519
- Hash
  - SHA-2
    - Length: 224, 256, 384, 512
  - SHA-3
    - Length: 224, 256, 384, 512
  - BLAKE
    - Length: 224, 256, 384, 512
  - RIPEMD-160
- MAC
  - CMAC
  - GMAC
  - HMAC

AUTOSAR Crypto API (ctd)

The following cryptographic algorithms or primitives should be supported by the Crypto Stack:

- Random Number Generation
  - Deterministic Random Number Generator (D
  - True Random Number Generator (T
- Symmetric Encryption
  - AES
    - K
    - CTR, GCM, OFB, CFB, XTS
  - PRE
    - ey Length: 128 bits
    - Modes: ECB, CBC, CTR, GCM, OFB, CFB, XTS
  - ChaCha12/ChaCha20
    - Key Length: 256 bits

All of them, I think

In particular, every trendy hipster algorithm and every mechanism anyone's ever heard of

- None of which are supported by the system hardware

AUTOSAR Crypto API (ctd)

CSM services use cryptographic algorithms that are implemented using cryptographic software or hardware modules – both are out of scope and not specified by AUTOSAR.

- Ah, that's OK then

AUTOSAR Crypto API (ctd)

Oh, we forgot to add PKI to the mix

- Let's fix that…

5.1.1.4.15 [SRS_CryptoStack_00031] The Crypto Stack shall provide an interface for parsing certificates

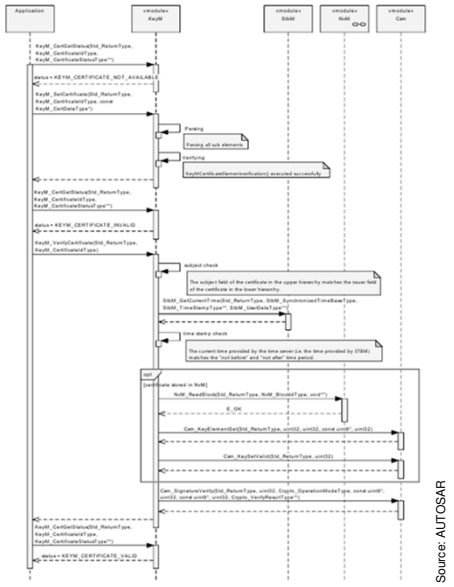
5.1.2.1.6 [SRS_CryptoStack_00111] The KeyM module shall support verification of certificates based on configured rules

5.1.2.1.7 [SRS_CryptoStack_00112] The KeyM module shall support retrieving arbitrary elements of a certificate

Source: AUTOSAR

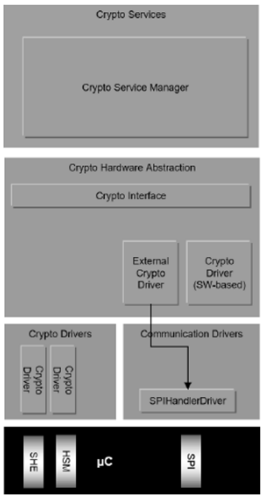
AUTOSAR Crypto API (ctd)

Specifically:



AUTOSAR Crypto Summary

Protocol Stack



The memory overhead of RTA-OSEK is:

Memory type	Overhead (bytes)
RAM	28
ROM/Flash	204

In addition to the RTOS overhead, each object used by an application has the following memory requirements:

Object	RAM Bytes	ROM Bytes
BCC1 task	0	36
BCC2 task	10	52
ECC1 task	28	60
ECC2 task	30	68
Category 1 ISR	0	0
Category 2 ISR	0	64
Resource	0	20
Internal Resource	0	0
Event	0	4
Alarm	12	60
Counter	4	44
Taskset (RW)	4	4
Taskset (RO)	0	4
Schedule	16	36
Arrivalpoint (RW)	12	12
Arrivalpoint (RO)	0	12

AUTOSAR Crypto Summary (ctd)

Other problems

- PKCS #11: Spec follows implementation
- CDSA, AUTOSAR: Here's a spec we dreamed up, someone else can figure out how to implement it

Spec is unclear, convoluted, ambiguous, difficult to follow, contradicts itself, requires reverse-engineering the thought processes of the author(s), …

- Far more so than the usual problems with standards

AUTOSAR Crypto Summary (ctd)

Unintentional humour

> **[SWS_Crypto_00013]** [The Crypto Driver may support all crypto primitives that are supported by the underlying hardware object.](SRS_CryptoStack_00098)

- What is the sound of no hands clapping?

Gratuitous reinvention of data formats…

> **7.2.5.2 Definition of ECC Key Material**
> **[SWS_Crypto_00189]**[Due to a lack of clear and efficient standard definition for ECC keys, key material for ECC is defined as binary information in the format definition of CRYPTO_KE_FORMAT_BIN_OCTET. The length of data depends on the assigned curve operation.
>
> Source: AUTOSAR

- ANSI X9.62 standardised ECC key formats twenty years ago!
- Everything (except AUTOSAR) uses the X9.62 format

AUTOSAR Crypto Summary (ctd)

… that are the reverse of the standard way of representing things

4.3.1 Integer-to-Octet-String Conversion

Input: A non-negative integer x, and the intended length k of the octet string satisfying:

$$2^{8k} > x.$$

Output: An octet string M of length k octets.

1. Let $M_1, M_2, \ldots, M_k$ be the octets of M from leftmost to rightmost.
2. The octets of M shall satisfy:

$$x = \sum_{i=1}^{k} 2^{8(k-i)} M_i .$$

- That's "big-endian" in math-ese

[SWS_Crypto_00190] ⌈ Public keys for NIST and Brainpool ECC curves are provided with their X and Y coordinates:

ECC Public Key = Point X | Point Y.

The points are stored in little endian format.

Source: AUTOSAR

- That's "the opposite of big-endian" in plain English
- Also, no handling of point compression

75

AUTOSAR Crypto Summary (ctd)

More gratuitous reinvention of 20-year-old industry standards

PKCS #15 v1.0: Cryptographic Token Information Format Standard

RSA Laboratories

April 23, 1999

[SWS_Crypto_00184]⌈ Asymmetric key material with identification is specified in

7.2.5.1 Definition of RSA Key Material

[SWS_Crypto_00185]⌈ For CRYPTO_KE_FORMAT_BIN_ RSA_PRIVATEKEY the

[SWS_Crypto_00186]⌈ The RSA public key in the format CRYPTO_KE_FORMAT_BIN _RSA_PUBLICKEY is provided as follows:

[SWS_Crypto_00187]⌈ The RSA public key in the format CRYPTO_KE_FORMAT_BIN _IDENT_RSA_PUBLICKEY is provided as follows:

[SWS_Crypto_00188]⌈ The algorithm identifier for RSA keys shall have the value

7.2.5.2 Definition of ECC Key Material

[SWS_Crypto_00189]⌈ Due to a lack of clear and efficient standard definition for

[SWS_Crypto_00190]⌈ Public keys for NIST and Brainpool ECC curves are provided with their X and Y coordinates:

[SWS_Crypto_00191]⌈ Private keys for NIST and Brainpool ECC curves are provided with their X and Y coordinates and an additional scalar:

[SWS_Crypto_00192]⌈ The public key information for ED25519 contains a point on

[SWS_Crypto_00193]⌈ The private key information for ED25519 contains a random

76

AUTOSAR Crypto Summary (ctd)

Left as an exercise for the reader

Description:	The KeyM shall provide interfaces for providing a proof for correctly programmed keys. A verification function can be used to provide a proof to the key server if the correct key is programmed and associated with a specific job.

Extensive mind-reading skills required to reverse-engineer the intent behind various requirements

The value of redirectionConfig is a bit coded value that is used to indicate, which of the input and output buffers are redirected. If the least significant bit (Bit #0 or 0x01) of redirectionConfig is set the primary input key and its element is redirected and the value of inputKeyId and inputKeyElementId must indicate the element that is used for input buffer instead of the inputPtr and its length. If Bit #1 is set, the secondaryInputBuffer is redirected to the secondary input key is set and the key and key elements must be set, and Bit #2 is used for the tertiary input key. Bit #3 is reserved for future use.
If Bit #4 is set the outputPtr is redirected to the output key element of the output key. Bit #5 indicates the redirection of the secondary output buffer to the secondary key and its key element. If a bit is set to 0 the input or output shall not be redirected to the associated Key Element.

Source: AUTOSAR

AUTOSAR Crypto Summary (ctd)

More mind reading

[SWS_KeyM_00099] [If a key was identified by its ID and either RequestDataPtr and RequestDataLength indicates data or KeyM_KH_Update() has returned E_OK and ResultDataPtr and ResultDataLengthPtr indicates data and the configuration /KeyMCryptoKey/KeyMCryptoKeyGenerationType is set to KEYM_STORED_KEY, then this function shall call Csm_KeyElementSet to provide the data to CSM. The key element ID is always 1 and the *KeyMCryptoKeyCsmKeyTargetRef* is used to identify the target key.

[SWS_KeyM_00100] [If a CryptoKey container was found and either RequestDataPtr and RequestDataLength provides data or KeyM_KH_Update() has returned E_OK and ResultDataPtr and ResultDataLengthPtr provides data and the configuration /KeyMCryptoKey/KeyMCryptoKeyGenerationType is set to KEYM_DERIVE_KEY, then the data shall be set to the key element CRYPTO_KE_KEYDERIVATION_PASSWORD. If the configuration value

Source: AUTOSAR

Zaphod felt he was teetering on the edge of madness and wondered whether he shouldn't just jump over and have done with it

— Douglas Adams

AUTOSAR Crypto Summary (ctd)

Ambiguities in the spec

Service name:	Csm_SignatureGenerate	
Syntax:	`Std_ReturnType Csm_SignatureGenerate( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr )`	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perform.
	dataPtr	Contains the pointer to the data to be signed.
	dataLength	Contains the number of bytes to sign.

Source: AUTOSAR

Does this sign a message, a hash of a message, or a formatted hash of a message?

- Expected operation is to sign a formatted hash, not the raw message or hash
- Key is associated with a job (= PKCS #11 handle), so it's probably signing a hash/formatted hash, not a message

AUTOSAR Crypto Summary (ctd)

Further ambiguities abound

8.3.4.4.2 Crypto_KeyGenerate
[SWS_Crypto_91007] ⌈

Service name:	Crypto_KeyGenerate
Syntax:	`Std_ReturnType Crypto_KeyGenerate( uint32 cryptoKeyId )`

- Which parameters?

8.3.4.5.1 Crypto_KeyDerive
[SWS_Crypto_91008] ⌈

Service name:	Crypto_KeyDerive
Syntax:	`Std_ReturnType Crypto_KeyDerive( uint32 cryptoKeyId, uint32 targetCryptoKeyId )`

Source: AUTOSAR

- Which derivation mechanism and parameters?

AUTOSAR Crypto Summary (ctd)

What are these things?

CRYPTO_ALGOFAM_ECCANSI	0x1e
CRYPTO_ALGOFAM_ECCNIST	0x19

- Aren't these the same thing?

CRYPTO_ALGOFAM_ECCSEC	0x1f

- SECG perhaps?

CRYPTO_ALGOFAM_FIPS186	0x21

- Which one? DSA, RSA, or ECDSA?

CRYPTO_ALGOMODE_PXXXR	0x09

- No idea…

Source: AUTOSAR

AUTOSAR Crypto Summary (ctd)

PKCS #11

```
C_Digest(hSession, data, dataLen, digest, digestLen);
```

AUTOSAR

The application, i.e., a runnable of a SWC, communicates with the crypto stack, i.e. the CSM, via RTE ports. For each configured job, the RTE generates a port named *{Job}_MacGenerate* with the client/server interface *CsmMacGenerate_{Primitive}()*. This port has one port defined argument value *Crypto_OperationModeType* with value *CRYPTO_OPERATIONMODE_SINGLECALL*. Thus, the SWC may invoke the MAC service by calling *CsmMacGenerate_{Primitive}* and provides all data required to perform the MAC request in one single call to the CSM. A BSW module or a CDD makes use of the MAC service by directly calling the C-API *Csm_MacGenerate()*. Here, the job to be processed has to be passed as input parameter.

Depending on the configuration of the applied job, the processing of the job will be asynchronous or synchronous. In this example, we assume that the job has to be processed synchronously, in the context of the caller. The CSM dispatches the requested MAC service to CRYIF by calling *CryIf_ProcessJob()*, passing the job as input parameter. CRYIF transfers the processing of the job to CRYPTO by calling *Crypto_ProcessJob()* which finally executes the crypto primitive configured in the job parameter. Note, that if different CRYPTO implementations are available, the function naming will differentiate by using vendorId (vi) and vendorApiInfix (ai). Thus, the call would be *Crypto_<vi>_<ai>_ProcessJob()*. Finally, CRYPTO stores the MAC in the memory space configured in the CSM job.

Source: AUTOSAR

AUTOSAR Crypto Summary (ctd)

The PKI portion in particular still needs some years of work

- Nine years so far for the rest of the spec

8.3.4.7.2 Crypto_CertificateVerify
[SWS_Crypto_00171] [

Service name:	Crypto_CertificateVerify

Syntax:	`Std_ReturnType Crypto_CertificateVerify(`
	`  uint32 cryptoKeyId,`
	`  uint32 verifyCryptoKeyId,`
	`  Crypto_VerifyResultType* verifyPtr`
	`)`

Description:	Verifies the certificate stored in the key referenced by cryptoValidateKeyId with the certificate stored in the key referenced by cryptoKeyId.

Source: AUTOSAR

Algorithm inputs: Certificate, issuing certificate

AUTOSAR Crypto Summary (ctd)

Algorithm inputs from PKIX PKI standard

6.1.1. Inputs

This algorithm assumes that the following nine inputs are provided to the path processing logic:

(a) a prospective certification path of length n.

(b) the current date/time.

(c) user-initial-policy-set: A set of certificate policy identifiers naming the policies that are acceptable to the certificate user. The user-initial-policy-set contains the special value any-policy if the user is not concerned about certificate policy.

(d) trust anchor information, describing a CA that serves as a trust anchor for the certification path. The trust anchor information includes:

(1) the trusted issuer name,

(2) the trusted public key algorithm,

(3) the trusted public key, and

(4) optionally, the trusted public key parameters associated with the public key.

(e) initial-policy-mapping-inhibit, which indicates if policy mapping is allowed in the certification path.

(f) initial-explicit-policy, which indicates if the path must be valid for at least one of the certificate policies in the user-initial-policy-set.

(g) initial-any-policy-inhibit, which indicates whether the anyPolicy OID should be processed if it is included in a certificate.

(h) initial-permitted-subtrees, which indicates for each name type (e.g., X.500 distinguished names, email addresses, or IP addresses) a set of subtrees within which all subject names in every certificate in the certification path MUST fall. The initial-permitted-subtrees input includes a set for each name type. For each name type, the set may consist of a single subtree that includes all names of that name type or one or more subtrees that each specifies a subset of the names of that name type, or the set may be empty. If the set for a name type is empty, then the certification path will be considered invalid if any certificate in the certification path includes a name of that name type.

(i) initial-excluded-subtrees, which indicates for each name type (e.g., X.500 distinguished names, email addresses, or IP addresses) a set of subtrees within which no subject name in any certificate in the certification path may fall. The initial-excluded-subtrees input includes a set for each name type. For each name type, the set may be empty or may consist of one or more subtrees that each specifies a subset of the names of that name type. If the set for a name type is empty, then no names of that name type are excluded.

Source: RFC 5280

AUTOSAR Crypto Summary (ctd)

Verification process in AUTOSAR is also missing a little detail

6. Certification Path Validation 71
   6.1. Basic Path Validation 72
        6.1.1. Inputs 75
        6.1.2. Initialization 77
        6.1.3. Basic Certificate Processing 80
        6.1.4. Preparation for Certificate i+1 84
        6.1.5. Wrap-Up Procedure 87
        6.1.6. Outputs 89
   6.2. Using the Path Validation Algorithm 89
   6.3. CRL Validation 90
        6.3.1. Revocation Inputs 91
        6.3.2. Initialization and Revocation State Variables 91
        6.3.3. CRL Processing 92

Source: RFC 5280

Description:	Verifies the certificate stored in the key referenced by cryptoValidateKeyId with the certificate stored in the key referenced by cryptoKeyId.

Source: AUTOSAR

AUTOSAR Crypto Summary (ctd)

Required checks are as follows

3. If a revocation list is available, all involved certificates shall be checked if they are listed in the CRL.
4. The subject field of the certificate in the upper hierarchy matches the issuer field of the certificate in the lower hierarchy.
5. The current time provided by the time server (i.e. by STBM) shall be greater than the “not before” and lower than the “not after” time value.

Source: AUTOSAR

PKIX’ 21 pages omitted

In particular, no checking of key usage, key IDs, constraints, policies, CA status, …

- Brake controller manufacturer can issue a code-signing certificate to allow firmware updates on the ECU(s)
- As long as something has signed something else somewhere, it’s valid

AUTOSAR Crypto Summary (ctd)

Cert status is determined statically

8.3.3.10 KeyM_CertGetStatus

[SWS_KeyM_00133] [The certificate submodule shall maintain the status of a certificate and provide the status on demand.
[SWS_KeyM_00134] [A certificate has the status KEYM_CERTIFICATE_VALID if it was parsed and verified completely against other certificates of the PKI. All certificates of the chain of trust are available and verified completely.

Source: AUTOSAR

- What if it's expired or been revoked in the meantime?

Uhhh… what?

6. The signature can be verified with the associated public key of the certificate referenced by *KeyMCertUpperHierarchicalCertRef*. For X.509 certificates, all critical extension fields shall be present.

What you said was so confused that one could not tell whether it was nonsense or not
— Wolfgang Pauli

AUTOSAR Crypto Summary (ctd)

Several sections can't be implemented in an interoperable manner, if at all

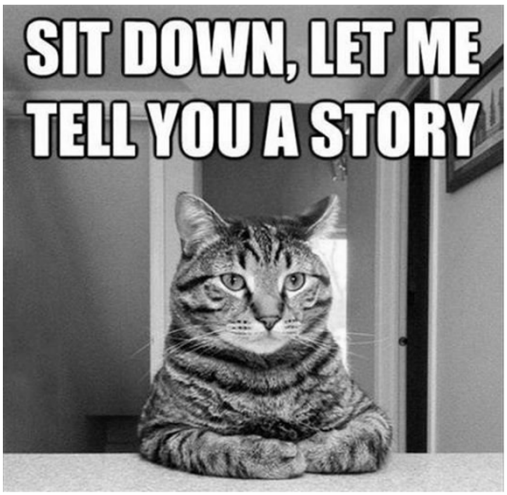
- Too vague or incomprehensible to implement unambiguously

No conformance profile(s)

- Which parts do you need to implement?
- Some = everyone does it differently
- All = see Apple, CDSA

AUTOSAR Crypto Summary (ctd)

How did things end up like this?



Reconsidering Automotive Security

What's our threat model?

AUTOSAR

- Here is a pile of crypto, whatever it counters is the threat
- a.k.a. "The Inside-Out Threat Model"

Reconsidering Automotive Security

Very popular in general, not just in automotive security

- A great many Internet security standards are based on the Inside-Out Threat Model

Reconsidering Automotive Security (ctd)

We control the environment

Manufacturer controls what's in the car

- The hardware
- The software
- The configuration

We don't need external attestation, identity management, certificates and CAs, etc

- Everything can be preconfigured at the factory

No need for public-key complexity, can use preconfigured symmetric keys

- Removes 95% of the complexity, and 95% of AUTOSAR API

LoRaWAN Security

Designed to solve the specific problem of device security in a practical manner

> Any crypto solution had better be based pretty exclusively on AES
>
> — Earlier slide

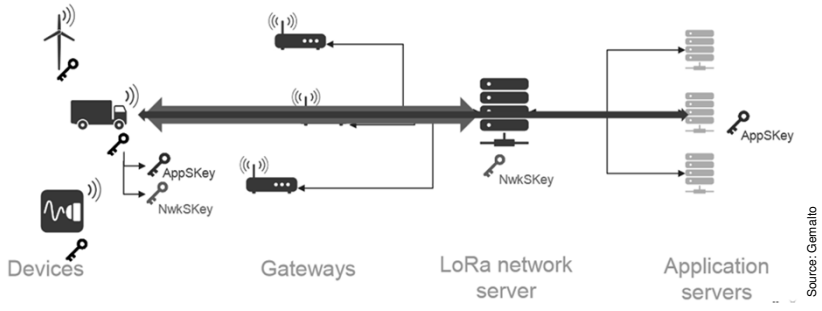
Manufacturer controls what's in the ~~car~~ device

- The hardware
- The software
- The configuration

We don't need […]

LoRaWAN Security (ctd)

Secures connections from devices at both the network and application level



- AES NwkSKey secures the network level
- AES AppSKey secures the application level

LoRaWAN Security (ctd)

Static activation: Device is provisioned at manufacture with NwkSKey and AppSKey

- Activation by Personalisation (ABP)

Dynamic activation: Device is provisioned at manufacture with EUIs and AppKey

- Over-the-Air Activation (OTAA) sends join command secured using the AppKey
- Servers derive AppSKey (application server) and NwkSKey (network server)
- All future traffic is encrypted and authenticated

Additional features to deal with replay attacks, ensure message uniqueness, etc

LoRaWAN Security (ctd)

We don't really need the NwkSKey vs. AppSKey distinction, just one will do

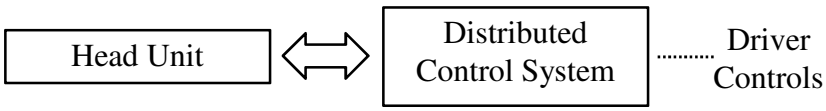
- Actually it's not clear what we need the AppSKey for either (see following slides)

Serves more as an example of what you can do if you start with a proper design

- Design follows threat model and functional requirements
- Create the most practical design that deals with as much of the threat model as possible

Reconsidering Automotive Security++

This is what a car really is



To attack parts of the distributed control system, the attacker has to have internal access to the vehicle

- At that point you're owned, with or without crypto

Weakness is access points like the head unit

- If an attacker can get into that, they're inside the device that controls the crypto
- Crypto becomes irrelevant

Reconsidering Automotive Security++

Difficult to identify a (non-artificial) situation where crypto would help against an actual real-world attack

- We need access controls, not crypto

This may be the only time you'll hear a crypto person tell you that the solution to a problem isn't to add more crypto

Reconsidering Automotive Security++ (ctd)

Isolate externally-accessible systems from control systems

You'll never make the head unit secure

- Attack surface is vast
- It needs to be a fully-functional media centre with everything enabled

Assume the head unit is pre-compromised

- Allow access to the control systems only via a carefully-controlled interface

Reconsidering Automotive Security++ (ctd)

Standard solution: Data diodes

- 1970s technology for dealing with classified government information

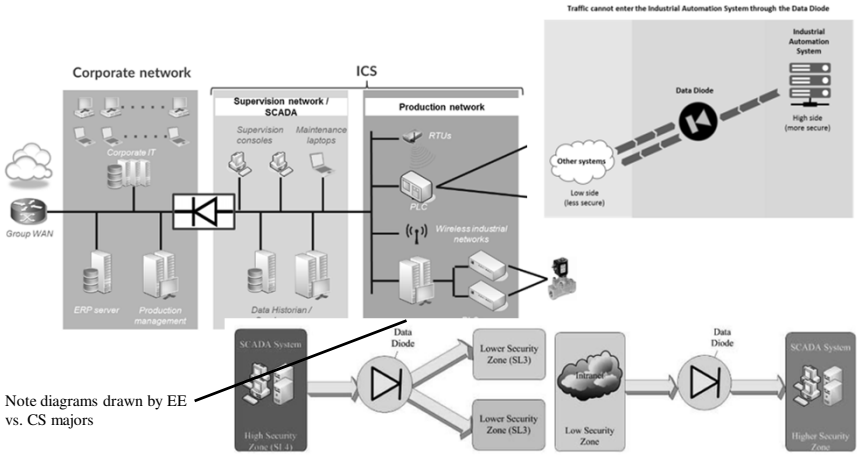
- Went mainstream in the 1990s when others started needing the capability

Low-tech hardware implementation

- 10mbps ethernet with transmit wire cut

Reconsidering Automotive Security++ (ctd)

Software implementations are widely used in SCADA environments



Sources: Schneider/GI-Jahrestagung/Brucon

Reconsidering Automotive Security++ (ctd)

More generally, implement the minimal interface necessary

Head unit mostly needs to display read-only telemetry

- Alongside music streaming, satnav, hands-free chat, video, …

Small number of outgoing commands, e.g. A/C, mirror control, are handled via carefully-checked interfaces

- There is no reason why your head unit should be able to actuate the brakes, gun the engine, lock the horn on, …
- (The Hell's Angels horn hack)

Some manufacturers already implement something like this

- VAG has a gateway from the high-speed control CAN bus to the low-speed convenience CAN bus

Reconsidering Automotive Security += 2

It's difficult to even identify a real-world situation where *any* sort of attack makes sense

Cybercriminals attack where the money is: Fraud, phishing, carding, ransomware, …

- No financial incentive to attack cars
- Ransom the playlist in your head unit?

Possible attack: Pay to allow your car to start

- Ransomware is a complex process typically involving BTC payments and taking days if not weeks
- Easier/quicker to just get a breakdown service/repair service/dealer to reflash your ECU

Reconsidering Automotive Security += 2 (ctd

Attacks on cars typically have to be local

- Fly a fast, powerful drone over the car at a matching speed
  - Drone is controlled from another vehicle close to the target
- Attack the head unit via Bluetooth
- Seize control of the car
- A black helicopter swoops in
- Jason Statham drops onto the roof
- …

This isn't really practical outside of movie plots

Reconsidering Automotive Security += 2 (ctd

Then what?

- Anything minor is mostly a nuisance attack, like keying the car
- Anything serious enough to get attention could be life-threatening

Liability goes from "phishing someone on the other side of the planet with low police interest" to "premeditated murder in the same police jurisdiction as the victim"

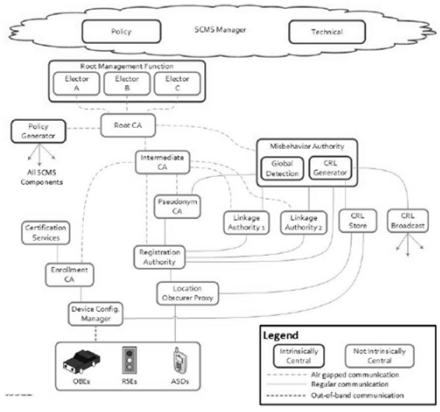
Need to threat-model/game-model what actually makes sense

- "Game theory" = modelling the actions of a rational player
- Maybe all we need is swift police action as a deterrent

But What About …?

V2V/V2I Communications

- That's an entirely different mess
- See IEEE 1609.2
- See also "IEEE 1609.2 and Connected Vehicle Security: Standards Making in a Pocket Universe", William Whyte

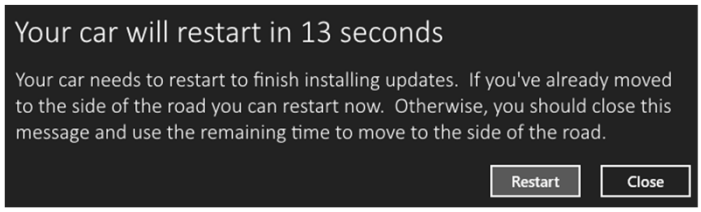


Source: William Whyte

But What About …? (ctd)

Firmware Updates

- Scenario: Travelling down the Autobahn at 200kmh…



- Even if you do it in the owner's garage, what if the upgrade bricks the car? Disables a safety-critical system?

Let's make it a dealer-only option…

- Dealer performs post-update tests to verify that all is OK

But What About …? (ctd)

What if the car dealer is the attacker?

- You're hosed, but this is a what-if scenario

Sign the firmware with a baked-in manufacturer key

- Standard practice in vast numbers of devices and systems

Conclusion

Automotive control systems security is a mess

- Security was never considered in the initial design because it wasn't needed
- Later, the Inside-Out Threat Model created non-solutions to non-problems

Toxic combination

- Automotive engineers don't know security
- Security geeks don't know automotive electronics

The real threat mitigation is via access control

- Crypto is just a distraction

Don't let crypto geeks work without adult supervision!