Advanced Topics in Distributed Systems

Topic 09

Security in Automotive Networks: Controller/Message Authentication

Manish Mahesh Dalvi

04-07-2019

Universität Stuttgart IPVS

Content

- Abstract and why we need security in automotive networks ?
- Software Security Protocols
- Keying techniques
- Software Security results
- Hardware Proposal - vulCAN
- vulCAN Setup and Explanations
- Conclusion

Automotive Networks Abstract

- Connected to the Internet over WiFi, BLE, 3G…
- Exposing information
- Various Proposals but cannot be tested due IP of Companies
- Possibility of numerous attacks

- Most Prominent Vehicular Buses
  - CAN-FD
  - FlexRay

Security Attack

CAN-FD

- CAN-FD made for higher bandwidth Demand
- Speed upto 8Mbps in data phase
- Supports upto 64bytes in frame
- BRS for switching baud rate

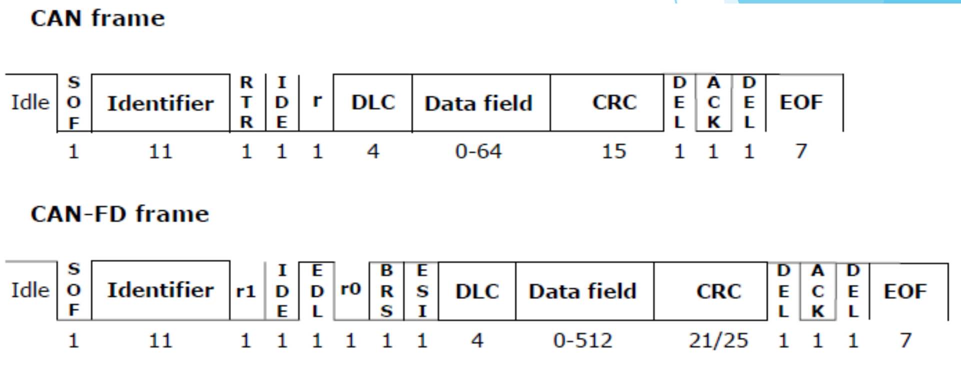


Figure 2: CAN frame versus CAN-FD frame

Universität Stuttgart IPVS

FlexRay

- Alternative to CAN, LIN
- Support upto 10Mbps
- Payload upto 254 bytes
- Time/Event triggered
- Static and Dynamic Data segment
- Network Idle Time at end Cycle for Synchronization
- Expensive, Safety critical applications

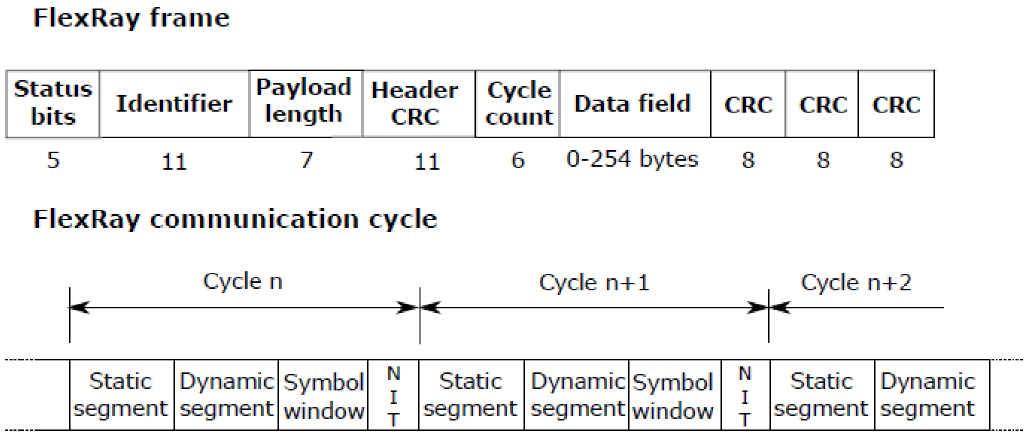
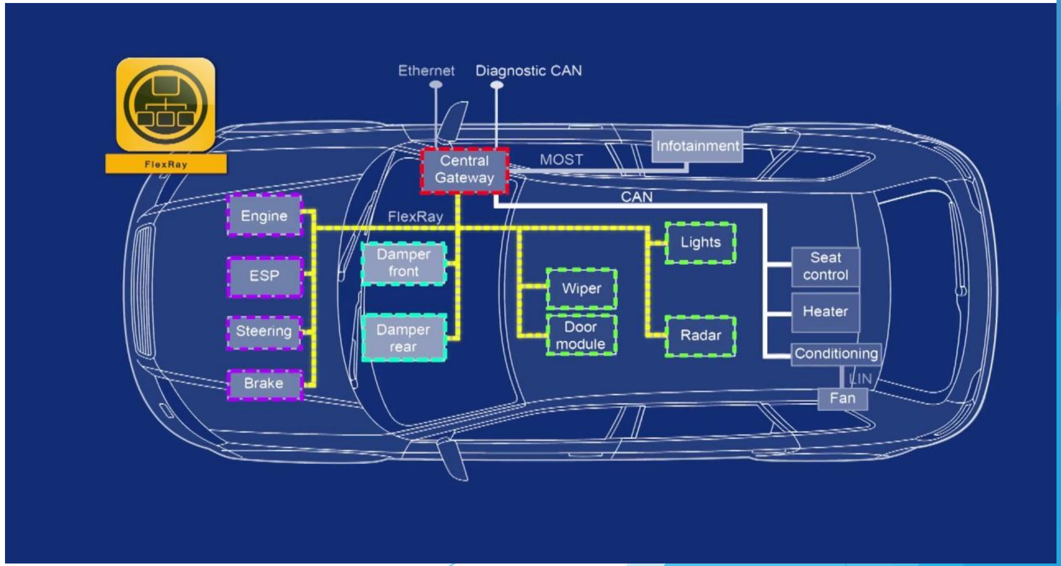


Figure 1: FlexRay frame and communication cycle



Types of Proposals

Software

Hardware

Software Proposals

- Voting Scheme
- TESLA
- CANAuth
- MaCAN
- LiBrA-CAN
- CaCAN

Voting Scheme

- Voting on Authenticity of the message
- All Nodes need to vote
- Additional delay
- Not suitable for real time

TESLA - Timed Efficient Stream Loss-tolerant Authentication

- Wireless Sensor network
- Authentication Tag sent separately
- All messages need to be buffered till Authentication Tag
- Adds a small delay of 1-10ms
- Memory consumption increased due to buffering of messages

CANAuth

- ID Oriented Key allocation
- ID assigned to each node
- Sharing keys with nodes which receive particular ID
- Too many ID’s can cause problem
- One tag per message is a good efficiency

MaCAN – Message authenticated CAN

- Shared keys between nodes
- Pair-wise key sharing
- Nodes of same hierarchy can be grouped for one tag
- Not sure how to share keys between nodes
- Trust level/hierarchy method not defined

LiBrA-CAN - Lightweight Broadcast Authentication CAN

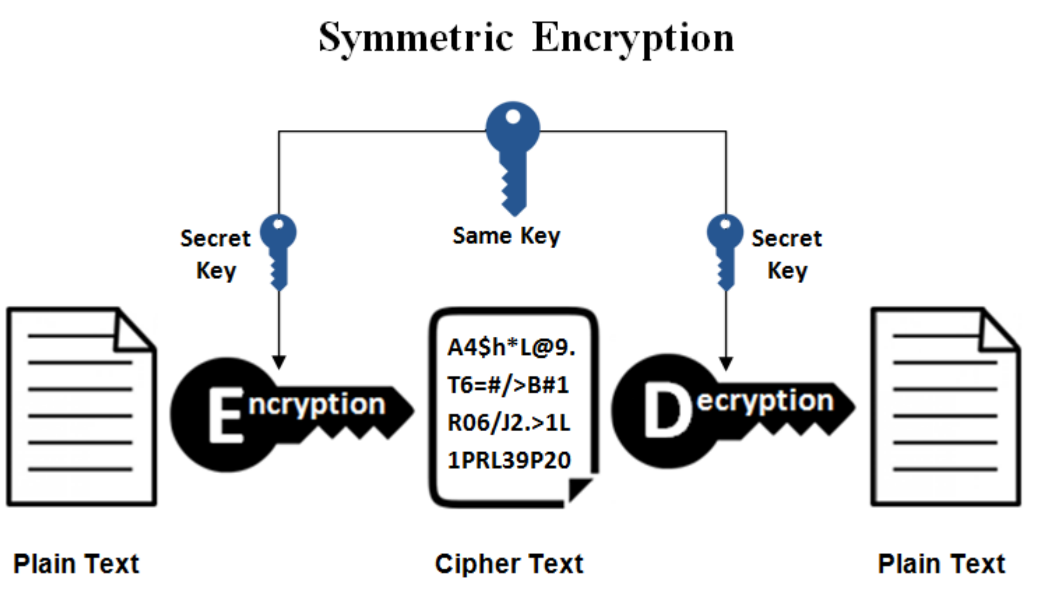
- Mix keys group wise than pairwise
- More computation power required
- Good security till corrupted nodes are in minority

CaCAN – Centralised authentication CAN

- Centralised authentication process
- Central node does the computation
- Central node failure causes system failure
- Central node failure means no authentication

Different types of Keying Techniques

- Single Authentication Keying
- Pairwise Keying
- Group keying
- Tesla-like Keying



Security level formulae

Total number of keys

$$\mathcal{K} = \binom{n}{g}$$

Number of keys stored in single node

$$\mathcal{K}_{\text{send}} = \binom{n-1}{g-1}$$

Tags intended for single receiver

$$\mathcal{K}_{\text{recv}} = \binom{n-2}{g-2}$$

Fraction of tags intended for single receiver

$$\mathrm{F}_{\text{recv}} = \binom{n-2}{g-2}\binom{n-1}{g-1}^{-1} = \frac{\mathcal{K}_{\text{recv}}^{\text{node}}}{\mathcal{K}_{\text{send}}^{\text{node}}},$$

Universität Stuttgart IPVS

Security level formulae

Size of tag for security of l bits

$$\mathrm{S} = \ell \binom{n-2}{g-2}^{-1} \binom{n-1}{k-1} = \ell \cdot \mathrm{F}_{\mathrm{recv}}^{-1}$$

Fraction of uncorrupted tags for a single receiver in m corrupted nodes

$$\mathrm{F}_{\mathrm{recv}}^{\mathrm{corr}} = \frac{\binom{n-2-m}{g-2}}{\binom{n-1}{g-1}}$$

Security level in case of m corrupted nodes

$$\ell^{\mathrm{corr}} = \mathrm{S} \cdot \mathrm{F}_{\mathrm{recv}}^{\mathrm{corr}}.$$

Security level diagrams

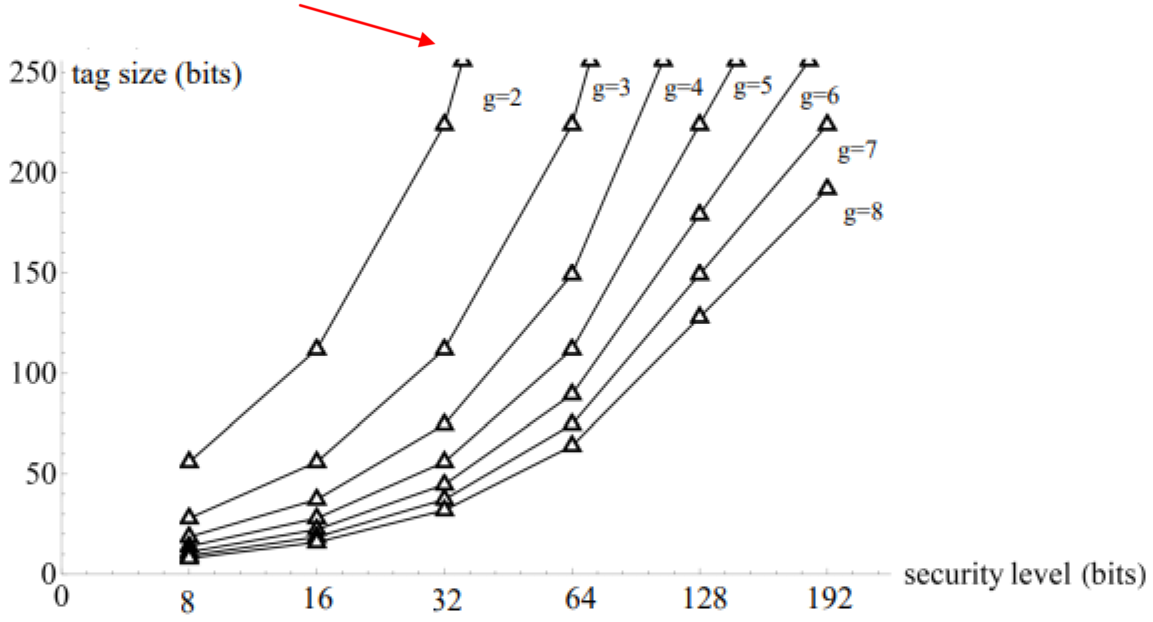


Figure 3: Tag size for a security level of 8 up to 192 bits with n=8 and g=2,3...8

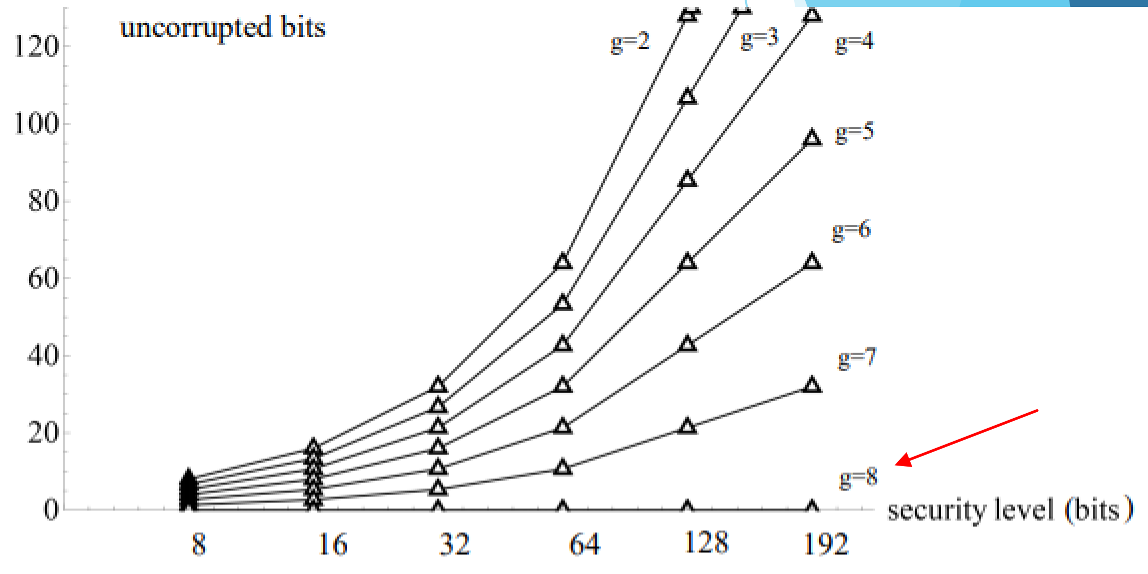


Figure 4: Uncorrupted bits in case of 1 corrupted node for a security level of 8 up to 192 bits with n=8 and g=2,3...8

Group size of 4 – 5 is a good trade off

Security level diagrams

- Total 8 nodes
- Group of 4 – Highest number of keys
- Group of 8 least – due to single authentication

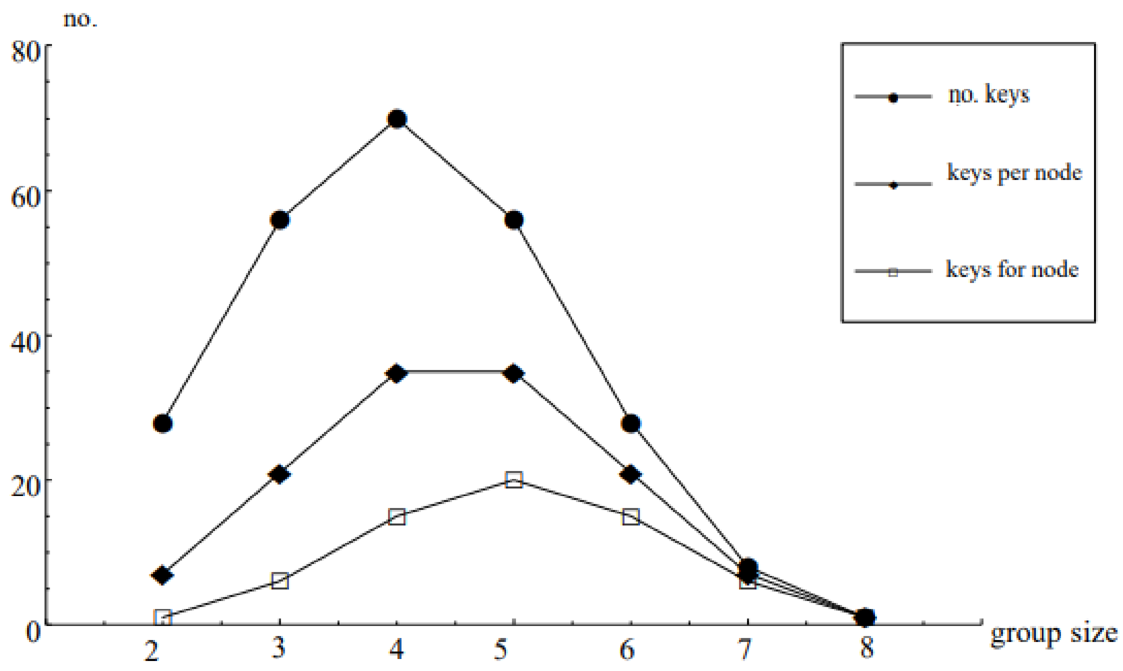


Figure 5: Comparison on the number of keys, number of computed tags and number of verified tags with n=8 and g=2,3...8

Experimental Setup

- Nodes grouped for Group keying
- Trust level/ Hierarchy
- Average of 2200 Frames/second/group

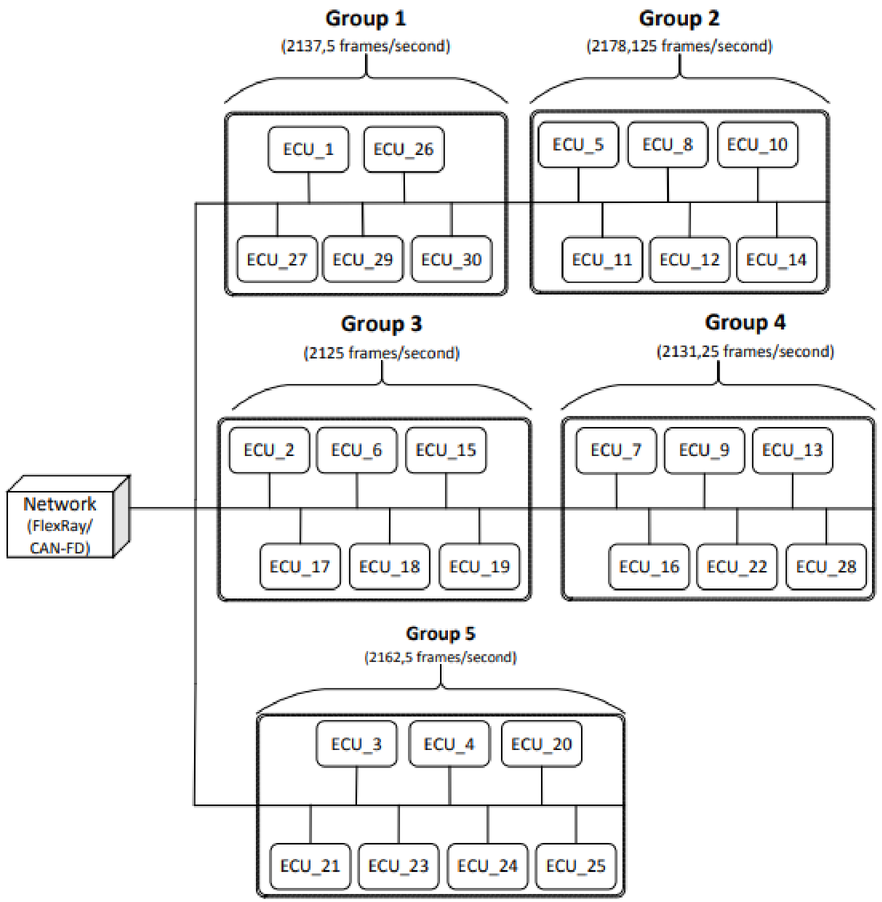


Figure 7: Grouping of ECUs into clusters

Comparative Results

Main Factors : Bandwidth and Computational Load

- Single Authentication key
  - Minimum load
  - Max increase in 128bits/frame
- Pairwise Keying
  - Number of Authentication tags = Number of ECU's
  - For 32-bit tags itself, the payload exceeds CAN-FD(64) and Flexray(256)
  - Not feasible

Tag size	No. of ECUs in network	Overhead [bits]	Overhead [bytes]
32	30	928	116
64	30	1856	232
128	30	3712	464
32	20	608	76
64	20	1216	152
128	20	2432	304
32	10	288	36
64	10	576	72
128	10	1152	144

Table 2: Frame overhead in case of pairwise keying

Universität Stuttgart IPVS

Comparative Results

- TESLA-like Keying
  - 10,20,80ms interval for authentication message
  - No significant increase in payload
  - Memory load increases exponential for Higher interval

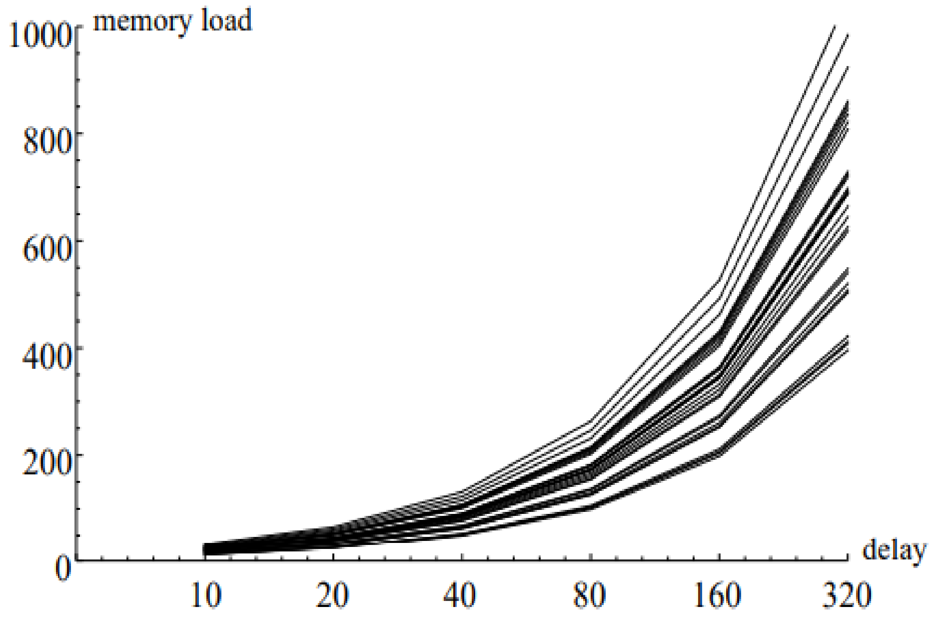


Figure 8: Dependence of memory load (in terms of frames waiting for authentication) with key release interval on each of the 30 ECUs with TESLA

Group Keying results

- Considered as 5 ECU’s instead of 5 clusters
- Maximum of 6 authentication tags
- Additional payload not significant
- Optimal compromise between security, payload and computational load

No. of clusters sharing a key	No. of receiver clusters	No. of added tags
2	1	1
	2	2
	3	3
	4	4
	5	4
3	1	3
	2	5
	3	6
	4	6
	5	6
4	1	3
	2	4
	3	4
	4	4
	5	4

Table 3: Number of tags added to a frame when applying *Group keying*

Results Table

- Pairwise keying does not fit into CAN-FD and Flexray payload
- Group keying has the highest computation payload – Handle with more hardware

Authentication protocol	Tag size [bits]	Max. payload [bits]	Max. payload [bytes]
Single authentication key	32	32	4
	64	64	8
	128	128	16
Pairwise keying	32	928	116
	64	1856	232
	128	3712	464
TESLA-like keying	32	32	4
	64	64	8
	128	128	16
Group keying	32	192	24
	64	384	48
	128	768	96

Table 4: Payload for the considered authentication protocols on all tag sizes

Authentication protocol	MIN comp. load [tags/s]	MAX comp. load [tags/s]	AVG comp. load [tags/s]
Single authentication key	1237.5	3290.625	2127.8125
TESLA-like keying	1237.5	3290.625	2127.8125
Group keying	3712.5	9871.875	6393.4375

Table 5: Computational load for the considered authentication protocols

Results Table

- Single Authentication key – Lowest bus load
- Group Keying - Highest busload

Authentication protocol	Recorded busload: CAN-FD [%]	Recorded busload: FlexRay [%]
Baseline	43.36	58.55
Single authentication key	48.97	58.57
TESLA-like keying 10 ms	64.97	71.61
TESLA-like keying 20 ms	57.40	65.89
TESLA-like keying 40 ms	53.62	61.56
TESLA-like keying 80 ms	51.73	59.41
Group keying (groups of 2)	68	58.57
Group keying (groups of 3)	82.21	58.57
Group keying (groups of 4)	70.51	58.57

Table 6: Recorded busload on CAN-FD and FlexRay

Next Solution

- vulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks

General idea about vulCAN

- Software component attestation, isolation
- Protect against network attackers as well as arbitrary code execution
- Sancus- Open source embedded protected module Architecture

MESSAGE AUTHENTICATION

Receivers get a strong guarantee that a message with specified ID and payload was indeed sent by a trusted sender component.

LIGHTWEIGHT CRYPTOGRAPHY

The use of public key cryptography is ruled out, for typical ECUs are severely constrained in computational power.

REPLAY ATTACK RESISTANCE

The authentication scheme is immune to replay attacks, even when large amounts of traffic were captured.

BACKWARDS COMPATIBILITY

VulCAN remains compatible with off-the-shelf CAN transceiver chips. Legacy applications continue to function.

REAL-TIME COMPLIANCE

VulCAN preserves real-time deadlines for safety-critical functionality such as brakes and steering while not under attack.

COMPONENT ISOLATION

Message processing algorithms and I/O operations are protected against an attacker with *arbitrary* untrusted code execution .

COMPONENT ATTESTATION

Critical software components are guaranteed to be isolated on specific ECUs without having been tampered with.

DYNAMIC KEY UPDATE

VulCAN supports secure key provisioning at runtime, and allows ECUs to be replaced by a distrusted automobile repair shop.

LEGACY ECU INTEGRATION

VulCAN makes it possible to transparently *shield* unmodified legacy ECUs as a transition measure.

Source : https://distrinet.cs.kuleuven.be/software/vulcan/

General idea about vulCAN

- TCB – Trusted Computing Base with minimal software
- Hardware enforced memory protection
- Isolate critical software components on ECU's
- Protect against Code Abuse attacks using return oriented Programming
- Making a untrusted I/O Interfaces into trusted
- vatiCAN and LeiA – AUTOSAR CAN Compliant authentication protocol used

Vulcanised CAN Components

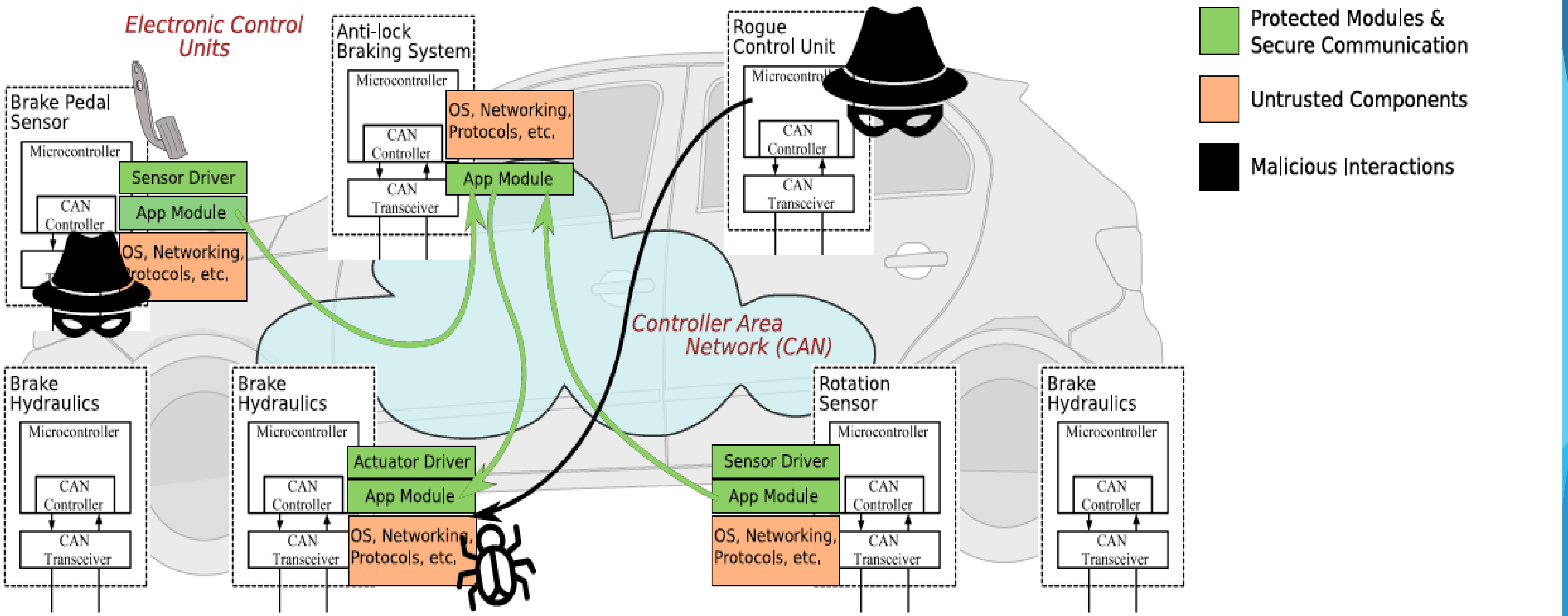


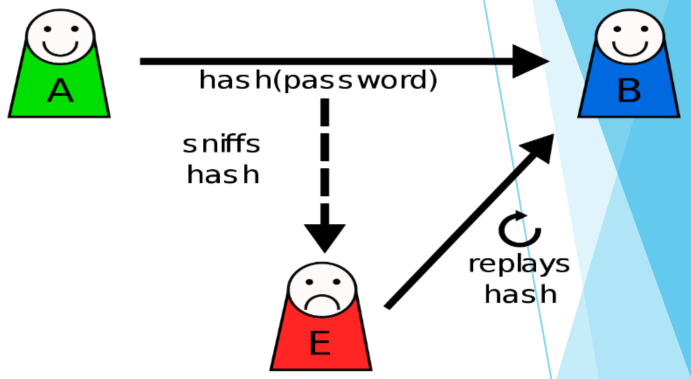
Figure 2: An example CAN network scenario to illustrate basic attacks and the security guarantees offered by our approach.

Outside Attacker

- Impersonate Protected component
- Arbitrary Message Manipulation
  - Successfully gained remote access
  - Broadcast own CAN message
  - Observe all traffic
  - Intentionally destroy/modify messages
  - Denial of Service out of scope
- Arbitrary Code Execution
  - All of software compromised except Sancus

Expectations from protocol

- Message Authentication
  - Message sent by Trusted sender component
- Lightweight Cryptography
  - Use lower computational power and storage space
- Replay Attack Resistance
  - Immune to replay attacks
- Backward Compatibility
  - Compatible with existing CAN transceivers



Source - https://en.wikipedia.org/wiki/Replay_attack#/media/File:Replay_attack_on_hash.svg

Protocols addressed

- Message Authentication and Lightweight Cryptography
  - Symmetric 128 bit key, grouping of IDs for one key
- Backwards Compatibility
  - Sending 2 messages – Normal and Authenticated message
- Replay Attack Resistance
  - Monotonically increasing counter

$$m = MAC(key_i, (i \mid p \mid c_i))$$

Protocols addressed

- Nonce Initialisation
  - Forbid replaying of previously authenticated messages
  - Short term Session keys
  - Counter overflow – fresh session key
- Nonce Resynchronization
  - Dealing with packet loss during sleep or heavy traffic
  - 16 bit nonce in extended CAN identifier field
  - Send error frame
  - Global Nonce generator to reset entire network nonce every few milliseconds

Expectations from System

- Real Time Compliance
  - Preserve real time deadline
- Component Isolation
  - Protection of message identity, key, authentication algorithms
- Component Attestation
  - No software tampering
- Dynamic Key Update
  - Secure key provisioning at run time
- Secure Legacy ECU Integration
  - Shield unmodified legacy ECU’s

System Requirements addressed

- Real Time Compliance
  - 128 bit hardware level encryption
  - Parallel computation of MAC on sender and receiver
- Component Isolation
  - Key in private data section of PMA
  - Message verified in Sancus PM
- Shielding legacy based ECUs
  - Sancus enabled gateway front of legacy ECU

Source - https://en.wikipedia.org/wiki/Black_box#/media/File:Blackbox3D-withGraphs.png

Universität Stuttgart IPVS

Component Isolation and Authentication

- Software Attestation and Key Provisioning
  - 3 Challenges
    - Integrity of critical distributed software components
    - Establish session keys over untrusted CAN bus
    - Replace Broken ECU’s in an untrustworthy automobile repair shop
  - Solution
    - Attestation Server – Has 128 bit Sancus key

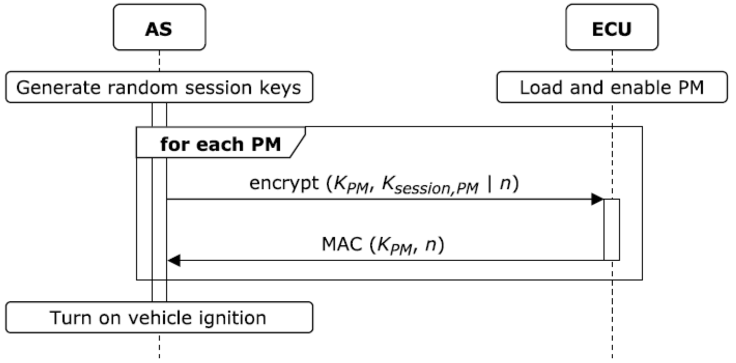


Figure 3: Load-time attestation and session key provisioning protocol between trusted Attestation Server (AS) and individual ECUs hosting PM software components.

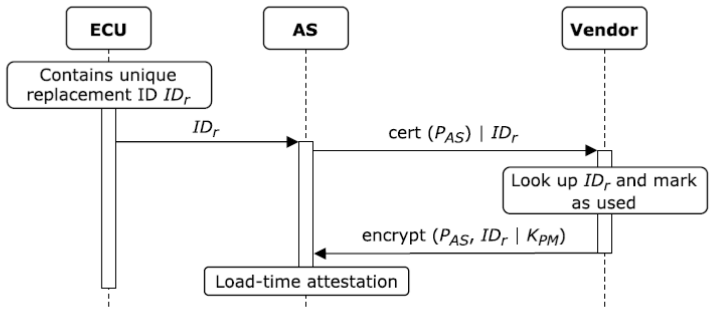


Figure 4: Protocol for integrating a new ECU into an existing control network. The in-vehicle Attestation Server (AS) is shipped with a certificate of its public key.

Security Analysis

- 2^128 brute force session keys – how long does it take ? Any idea ?
- 2^64 attempts for MAC
- Very small time frame since session keys keep changing
- Physical attackers out of scope
- SPONGEWRAP hash function

Setup

Figure 6: Hardware-in-the-loop application scenario with original instrument clusters and Sancus-enabled ECUs.

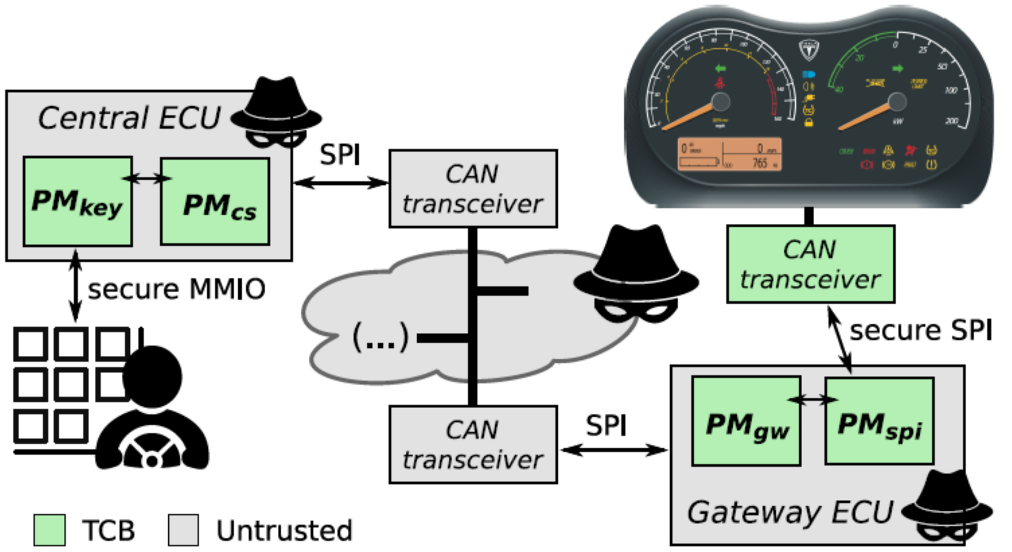


Figure 7: Schematic of the demo scenario depicted in Fig. 6.

Experimental Evaluation

- Hardware assisted Cryptography of SHA-3 reduces 47600 cycle to 4222 cycle

Table 1: Overhead to send an (authenticated) CAN message with/without Sancus encryption and software protection.

Scenario	Cycles	Time	Overhead
Legacy (standard ID)	8,135	0.41 ms	–
Legacy (extended ID)	9,620	0.48 ms	18%
vatiCAN (extrapolated†)	58,948	2.95 ms	625%
Sancus+vatiCAN (unprotected)	15,570	0.78 ms	91%
Sancus+vatiCAN (protected)	16,036	0.80 ms	97%
Sancus+LeiA (unprotected)	18,770	0.94 ms	131%
Sancus+LeiA (protected)	19,211	0.96 ms	136%

† Inferred from the observed Sancus+vatiCAN timings by replacing the hardware based MAC computation cycles with the reported Keccak SHA-3 computation cycles.

Experimental Evaluation

- Round trip time of Original message and Authentication Message

Table 2: Round-trip (ping-pong) time intervals.

Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated†)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%

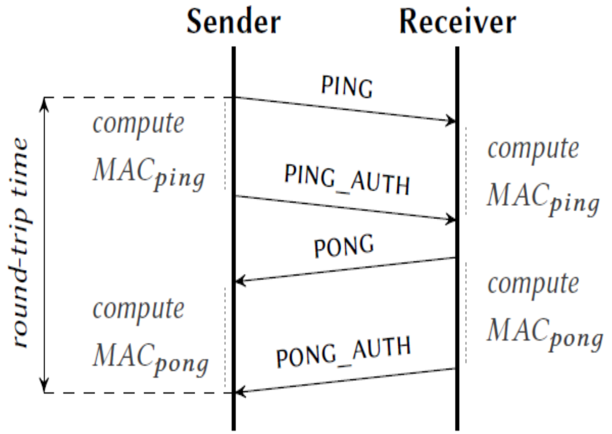


Figure 5: Round-trip time experiment timing overview.

Conclusions

- Software Proposal Group Keying
  - Cheaper, easy to implement, more backward compatible
  - Slower, can be made faster using Hardware encryption
- vulCAN
  - Faster encryption, Real time deadlines met better, more security
  - Lesser backward compatible
  - More things to change by a manufacturer
  - Expensive

Finally it depends on the purpose and there has to be a trade off made between Hardware and Software costs !

Thank you !

Questions ?