Applications of pairing-based cryptography on automotive-grade microcontrollers

Tudor-Sebastian Andreica Bogdan Groza Pal-Stefan Murvay

Faculty of Automatics and Computers,
Politehnica University of Timisoara,
Romania

September 18, 2018

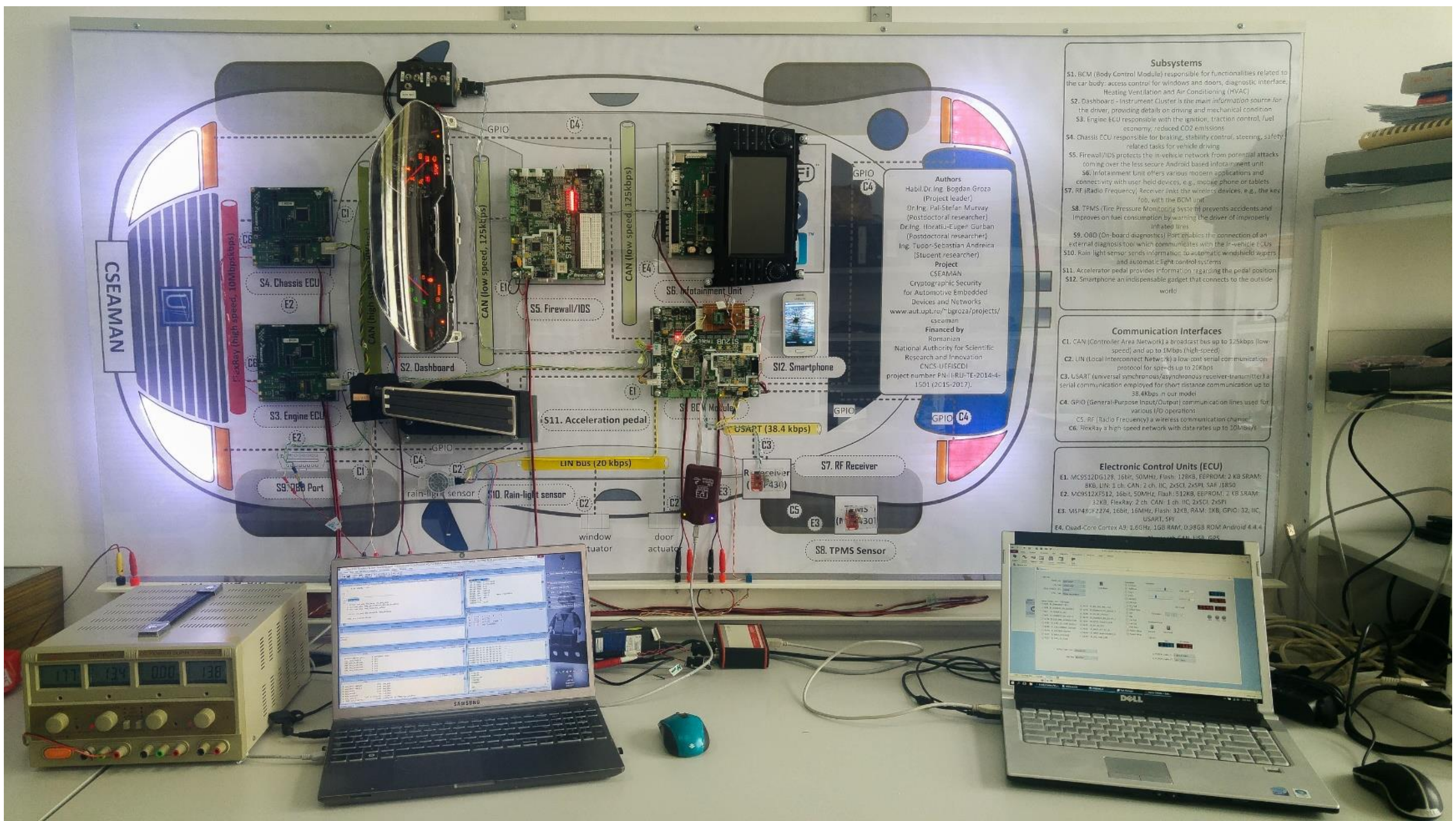
Research funded by

PRESENCE (PRivacy-Enabled, SEcured iNteractions between vehiCles and smart Electronic devices) young teams research grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, PN-III-P1-1.1.-TE-2016-1317 **(2018-2020)**

http://www.aut.upt.ro/~bgroza/presence.html

a sequel of CSEAMAN - Cryptographic Security for Automotive Embedded Systems and Networks, young teams research grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project number PN-II-RU-TE-2014-4- 1501 **(2015-2017)**.

http://www.aut.upt.ro/~bgroza/cseaman.html



Content of this presentation

I) What are pairings and what crypto-applications do they enable

II) How can they help in automotive-based scenario

III) Practical results from our work

I) Bilinear pairings in brief

- The bilinear pairing is a function $e: G_1 \times G_2 \rightarrow G_T$ having the following properties

1) bilinearity

$$\forall P, Q, R, \qquad e(P+Q, R) = e(P, R)e(Q, R), \quad e(R, P+Q) = e(R, P)e(R, Q)$$

2) non-degeneracy (means that it is nontrivial)

$$\forall P \neq \mathrm{O}, \exists Q \; s.t. \; e(P, Q) \neq 1$$
$$\forall Q \neq \mathrm{O}, \exists P \; s.t. \; e(P, Q) \neq 1$$

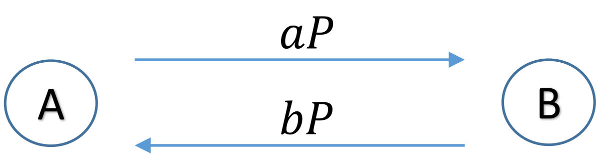
3) efficiently computable (means that it practically usable)

- By bilinearity it immediately follows

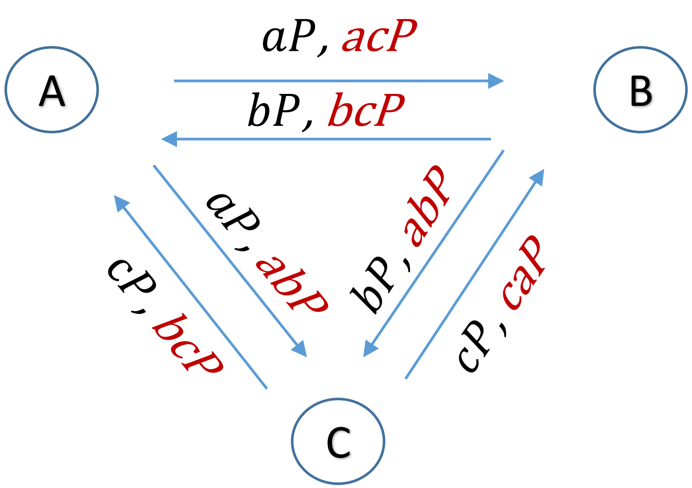
$$e(nP, Q) = e(P, Q)^n = e(P, nQ)$$

Exemplary application I – Tripartite Diffie-Hellman Key Exchange

- The Diffie-Hellman key-exchange is one of the pillars of Internet security (e.g., SSL/TLS, IPSec, SSH, etc.)
- Exchanging a key, i.e., abP, between 2 parties is easy

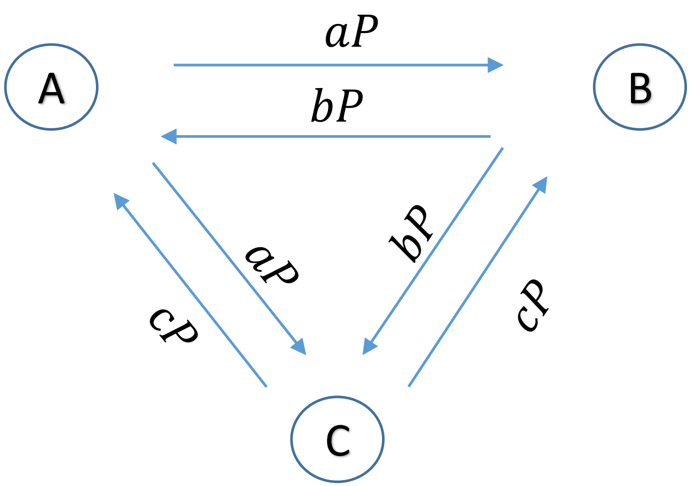


- Extending this between 3 parties is easy, but not necessarily efficient as sending only each party's share (i.e., aP, bP, cP) is not enough for computing the common key (i.e., $abcP$)



Pairings help by sending single value from each party

- Tripartite Diffie-Hellman, due to Joux'04



- Common key, i.e., , recovered by each party as

$$\text{A}: \; e(bP, cP)^{a} = e(P, P)^{abc}$$

$$B: \; e(aP, cP)^{b} = e(P, P)^{abc}$$

$$C: \; e(aP, bP)^{c} = e(P, P)^{abc}$$

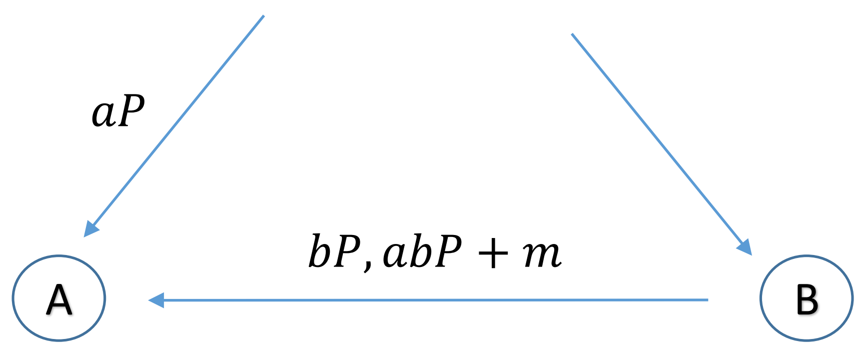
Exemplary application II – ID-based Encryption/Signatures

- Conventional public key encryptions/signatures require a digital certificate
- Shamir'84 introduces the term ID-based cryptosystem in which the identity of a party serves as the public key
- This cannot be achieved in a straight-forward way, e.g., a Diffie-Hellman based scenario:

A's public key is retrieved by applying a public "hash" function over the identity (user name)

user name: "*Alice*"

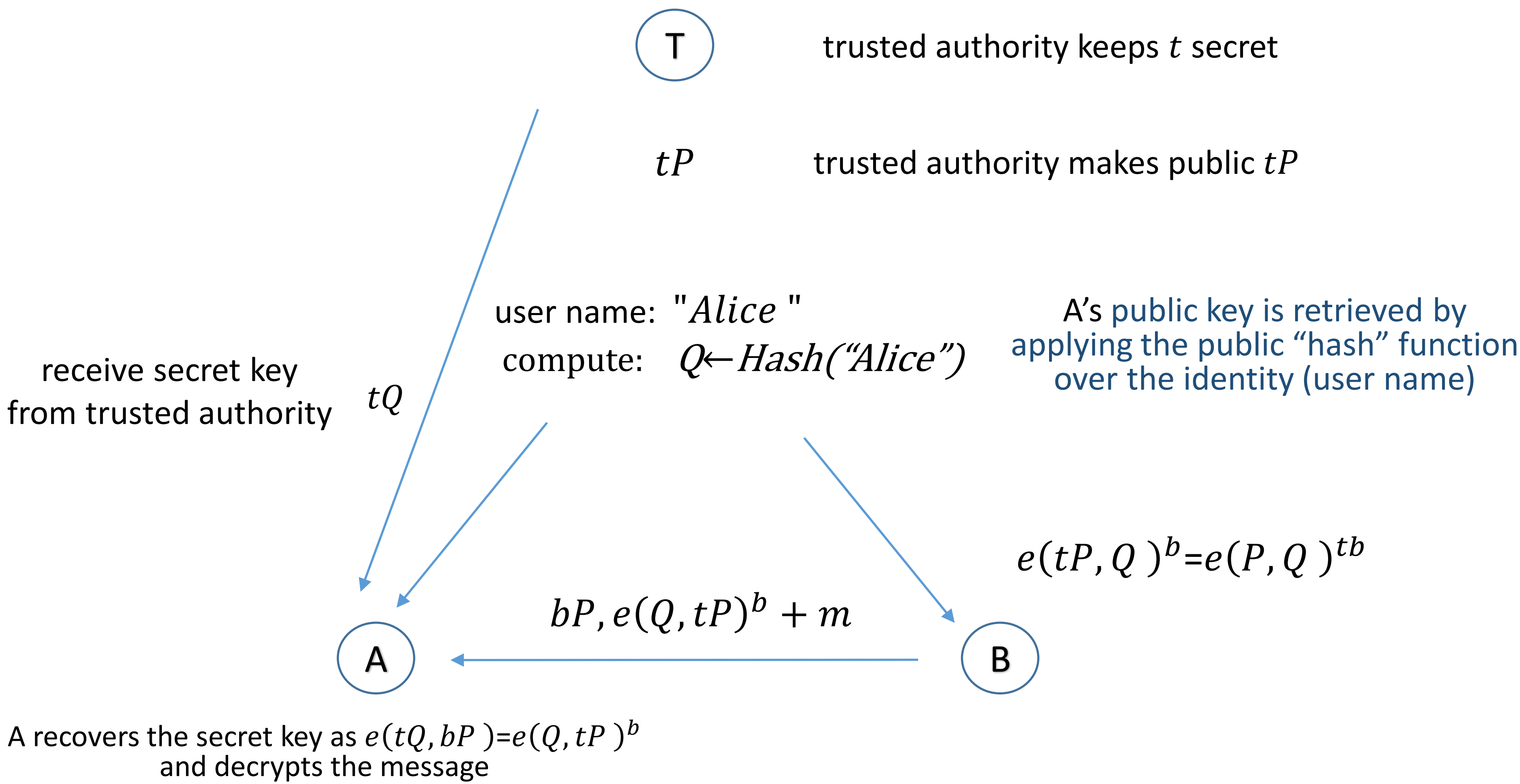
compute: $aP \leftarrow Hash(\text{“Alice”})$



but A cannot recover his private key, i.e., a, since the discrete log is intractable, and hence cannot decrypt

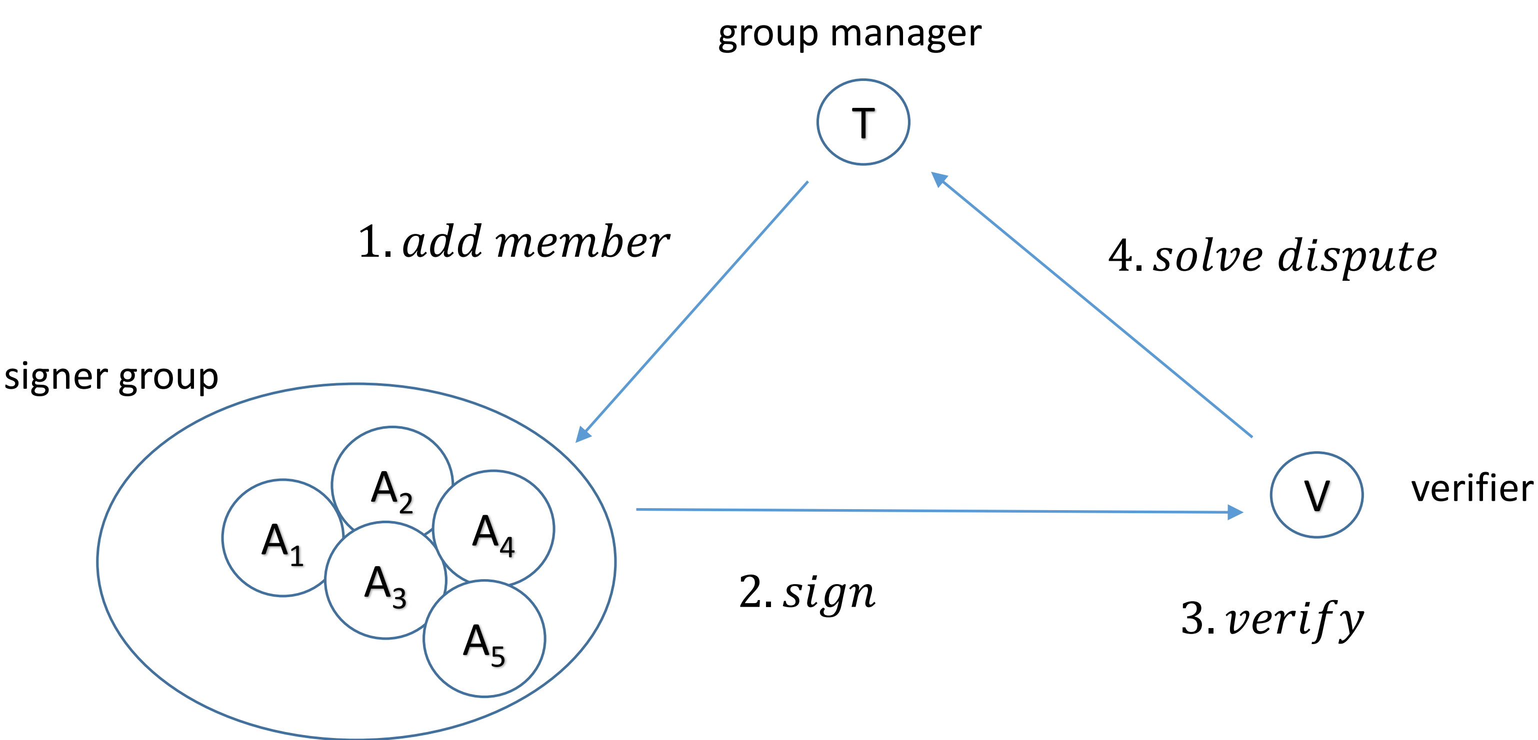
Pairings help in constructing identity-based encryption

- While ID-based signatures were proposed by Shamir in '86, ID-based encryption remain unknown until the use of pairings Boneh&Franklin '01
- With pairing it turns out to be quite trivial to derive identity-based encryption



Exemplary application III – Group Signatures

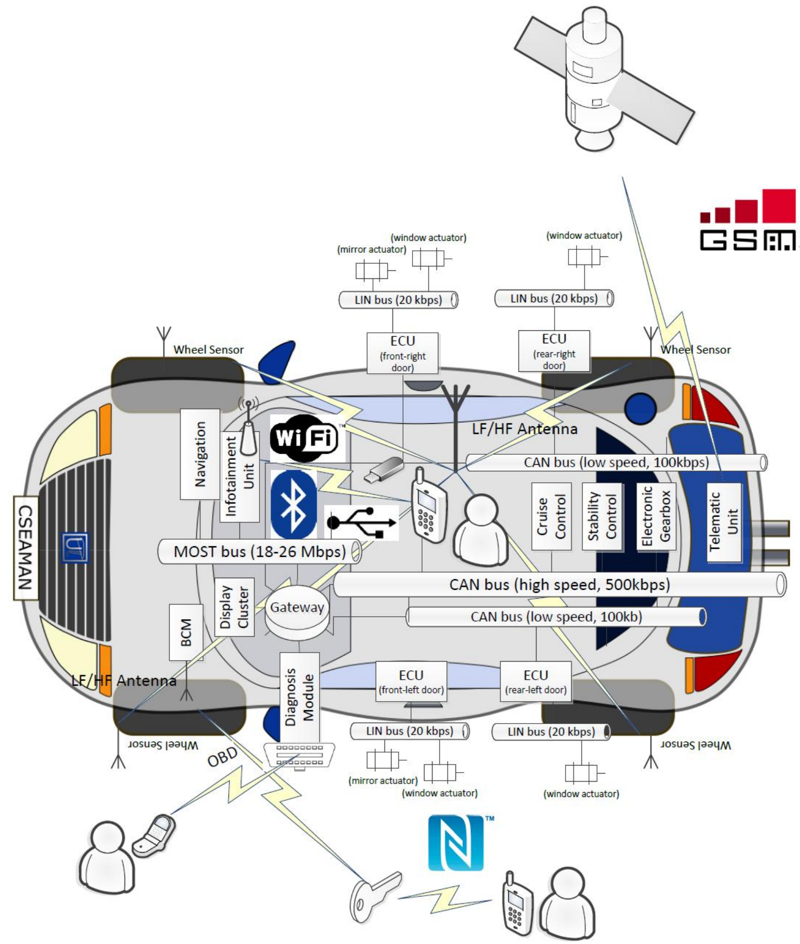
- Introduced in ‘91 by Chaum & Heyst
- One of the first practical scheme by Boneh, Boyen & Sacham’04 is based on pairings
- Concept:
  - any member of a group can sign a message (soundness & completeness)
  - the individual signer cannot be traced (anonymity)
  - the group manager can link the signature to a particular signer (traceability)
  - coalition of members cannot forge the signature of another member



II) Addressed scenarios: modern vehicle interconnectivity

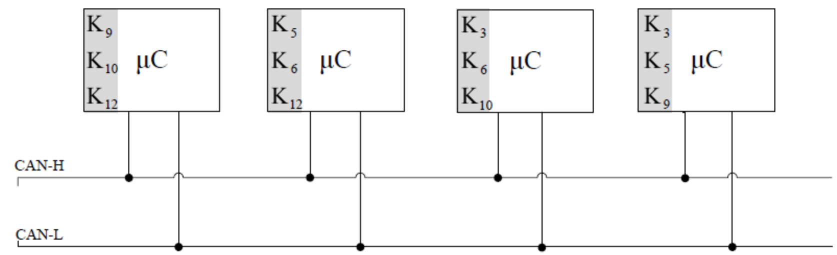
Vehicles evolved from mechanical devices into complex electro-mechanical systems loaded with software

- 100+ ECUs
- > 10 million lines of code
- Electronics + software
  =40% production cost
- 5-7 busses on various technologies
  - CAN,
  - FlexRay
  - BroadRReach (Ethernet),
  - LIN,
  - MOST, etc
- Several wireless interfaces
  - Bluetooth, WiFi, 4G



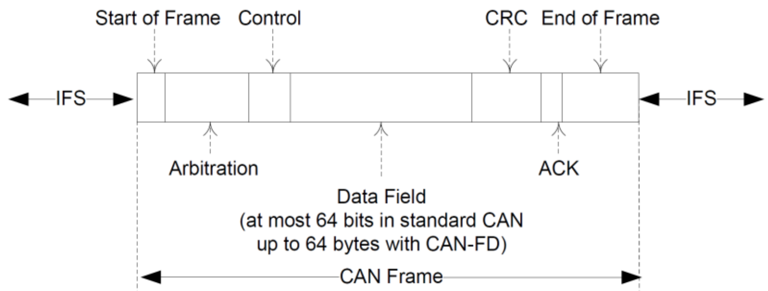
Challenge I - Assure security on in-vehicle buses

- CAN (Controller Area Network)
    - fault tolerant, low-speed version, max 125kbps (ISO11898-3)
    - high-speed , max 1Mbps (ISO11898-2)
    - CAN-FD (Flexible Data-Rate), max 2.5 Mbps
    - Payload size: 64 bits
    - CAN-FD extends this to 2.5Mbps, and 64 byte payload



- FlexRay
    - Fulfill communication req. of X-by-Wire
    - fault tolerant, high-speed, deterministic, max 2ch at 10Mbps

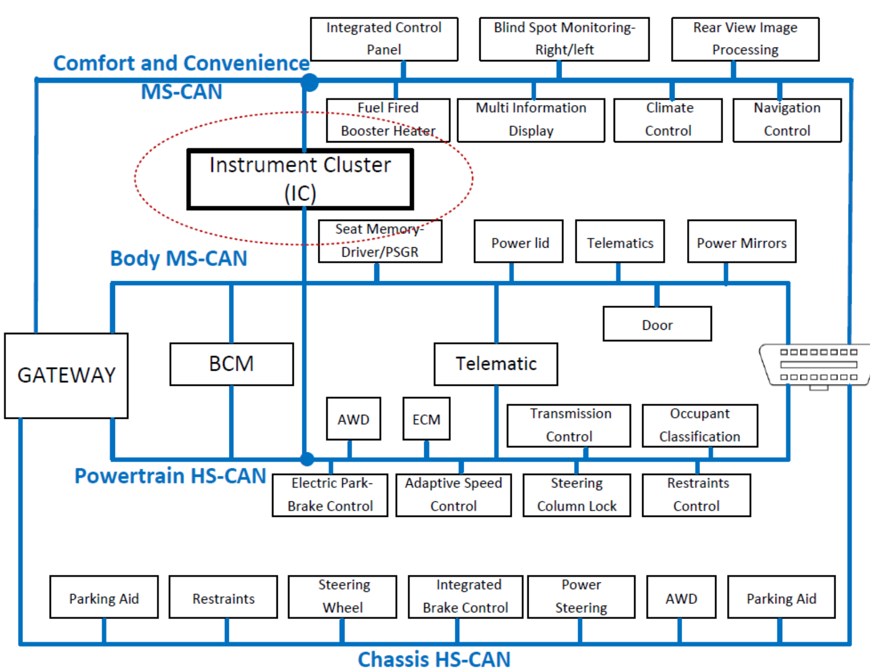
- LIN (Local Interconnect Network)
    - Low cost serial communication interface
    - based on a master-slave architecture
    - Connect peripheral sensors and actuators - max 20 kbps



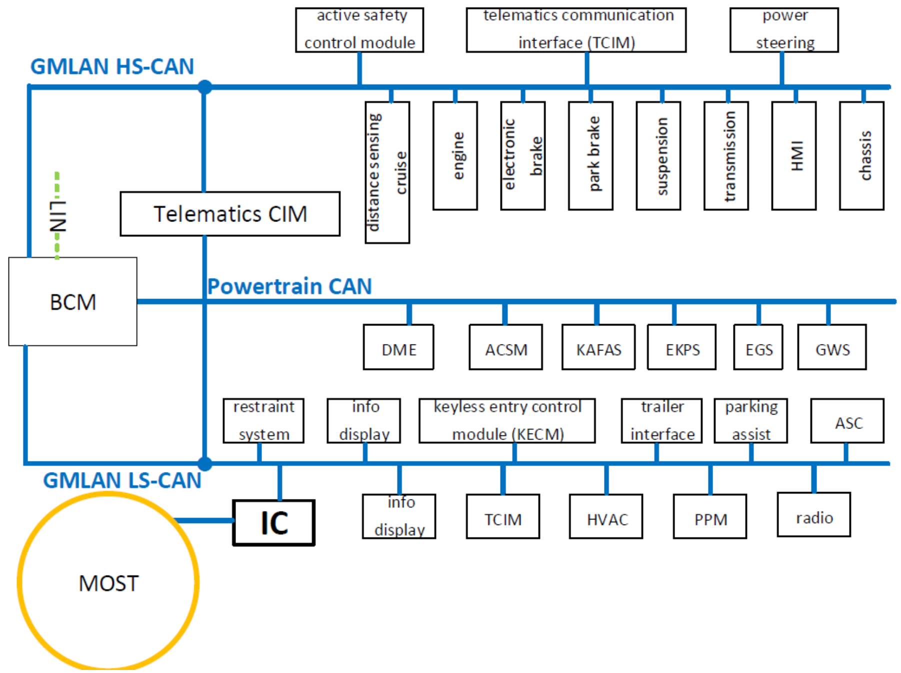
- None of these communication layers has any kind of security except for std. CRC codes

Network topologies

- No standardization, each manufacturers has it’s own type of network topology
- Result: security becomes even harder to assure



2014 Range Rover Evoque, redraw after C. Miller and C. Valasek [16]



2015 Cadillac Escalade AWD, redraw after C. Miller and C. Valasek [16]

- powertrain system
- comfort and convenience systems

- low speed CAN (LS-CAN) - body, comfort, ADAS and multimedia systems, etc.
- MOST network

Embedded platforms behind in-vehicle ECUs

Proposed µC by the first five suppliers of automotive semiconductor market: Renesas, Infineon, STMicroelectronics, Freescale and NXP

- 32bit architecture, with two or three cores
- clock speeds in 100-160MHz range
- Architecture with different cores with different clock speeds
- Multi-core architectures motivated by the double role (main controller for body functions and network gateway)

- The marketed controllers targets the higher end cars class due to:
  - high number of CAN ch. (between 6 and 8)
  - high number of LIN ch. (between 10 to 18)
  - presence of FlexRay and ETHERNET

Generally, sufficient computational resources for implementing standard cryptographic functions (but this also depends highly on real-time constraints)

Manufacturer	CPU characteristics	Peripherals microcontroller
Renesas	RH850 F1H CPU: 2 X RH850G3 32bit, 120 MHz Program Flash: 6MB EEPROM: 64 KB SRAM: 576 KB	I/O Port: 218; LIN Master: 12 ch. LIN/UART: 6 ch.; CAN: 7 ch. FlexRay: 1 ch.;Ethernet: 1 ch. HW Security Module (ICU-M)
Freescale	MPC564xB-C family MPC5646C CPU: e200Z4, 32bit, 120MHz e200Z0, 32bit, 80MHz Program Flash: 3 MB EEPROM: 64 KB SRAM: 256 KB	GPIO: 199; LIN: 10 ch. CAN: 6 ch.; FlexRay: 1 ch. Ethernet: 1 ch. Secure key storage AES-128 en/decryption, ECB/CBC Authentication with AES-128 CMAC SHE Secure boot protocol TRNG and AES-128 based PRNG
Freescale	MPC5748G family CPU:2x e200Z4,32b,160MHz 1 x e200Z2, 32bit, 80MHz Program Flash: 6 MB Data Flash: 192 KB SRAM: 768 KB	LINFlex: 18 ch. CAN: 8 ch. (CAN FD support) FlexRay: 1 dual-channel FlexRay Ethernet: 2 ch. One Secure Digital HW Controller
Infineon	XC2200 family XC2299H-200FxL, scalable 16/32bit,100 MHz Program Flash: 1.6 MB SRAM: 112 KB	GPIO : 150; LIN/UART : 10 ch. CAN: 6 ch.; FlexRay: 2 FlexRay Nodes Ethernet: none
STMicroelectronics	SPC56ECxx family SPC56EC74L7 CPU: e200Z4d, 32bit, 120MHz e200Z0h, 32bit, 80MHz Program Flash: 3 MB Data Flash: 64 KB SRAM: 256 KB	GPIO : 199; LIN/UART: 10 ch. CAN: 6 ch.; FlexRay: 1 dual channel FlexRay Ethernet: 1 ch. Cryptographic Services Engine (CSE), AES-128 en/decryption, CMAC auth., Secured device boot mode

source: Groza, Bogdan, Horatiu-Eugen Gurban, and Pal-Stefan Murvay, **Designing security for in-vehicle networks: a Body Control Module (BCM) centered viewpoint** , The 2nd International Workshop on Safety and Security of Intelligent Vehicles (SSIV 2016, affiliated to DSN 2016)

What are current standards in-vehicle security saying?

- MAC based security, possibly with truncated MACs may be enough
- Still, a key needs to be shared between nodes

AUTOSAR

Specification of Secure Onboard Communication
AUTOSAR CP Release 4.3.1

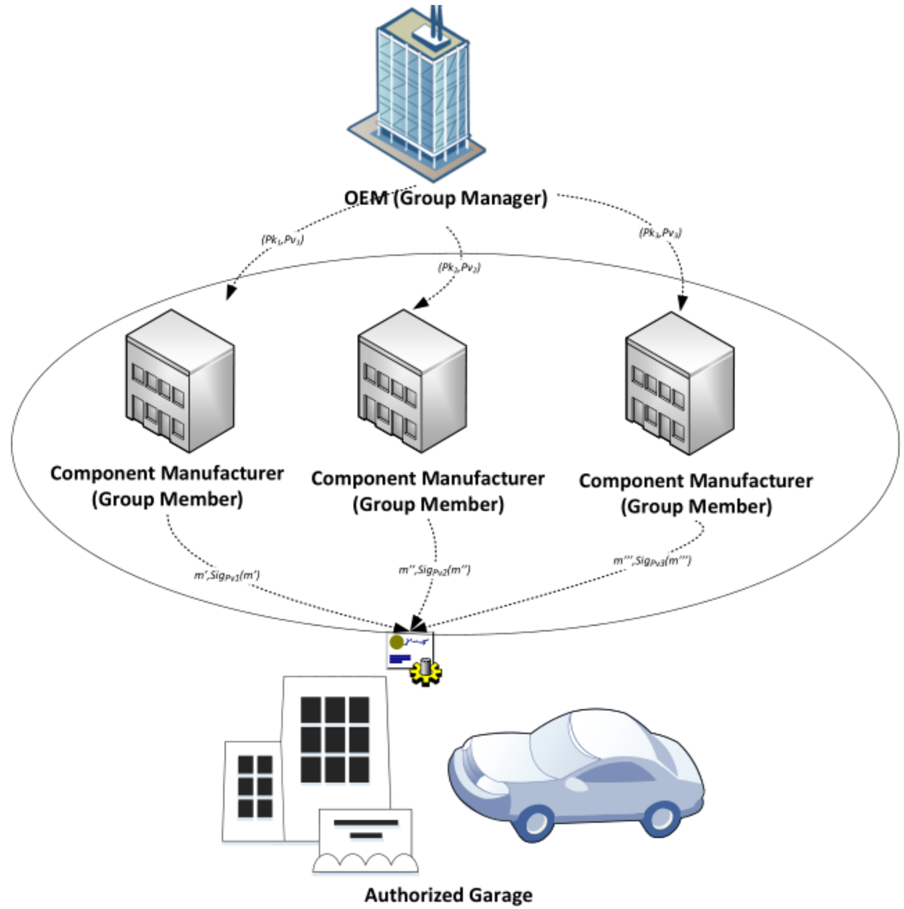
Parameter	***Configuration value***
Algorithm	CMAC/AES-128
Length of Freshness Value (parameter `SecOCFreshnessValueLength`)	Not Specified
length of truncated Freshness Value (parameter `SecOCFreshnessValueTxLength`	8 bits
length of truncated MAC (parameter `SecOCAuthInfoTxLength`)	24 bits

How PBC may help?

- More efficient key exchange between nodes, e.g., Tripartite Diffie-Hellman (saves bandwidth which is critical for in-vehicle buses)

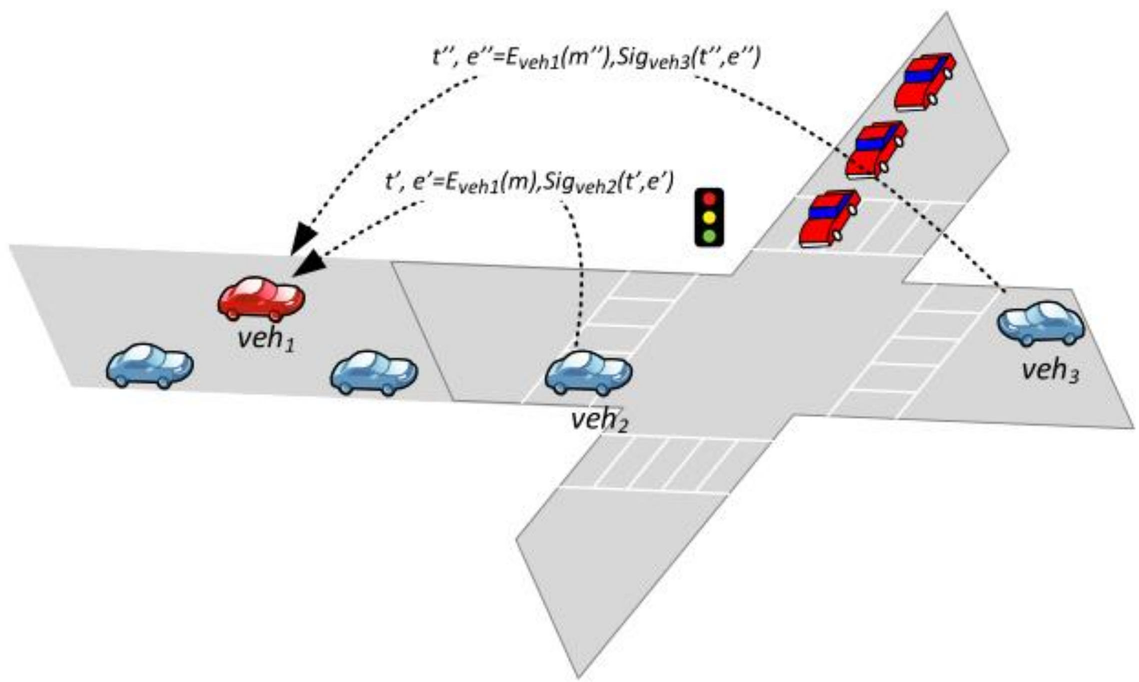
Challenge II – Security for software updates

- Group signatures may help as software may be signed by various component manufacturers
- The OEM as group manager can trace the signature back to the manufacturer
- The component manufacturer may remain anonymous for the car or the authorized garage which are still able to verify the signature



Challenge: security for V2V communication

- V2V outlined as potential application scenario even in the original Boneh-Boyen-Sacham '04 paper on short group signatures:
  - E.g., group signatures are suggested for preserving privacy of the signers BBS'04
- Identity based signatures may have a distinct application: retrieving public keys from public information, e.g., license plates
- License plates may serve as addition evidence that a car is present on-site

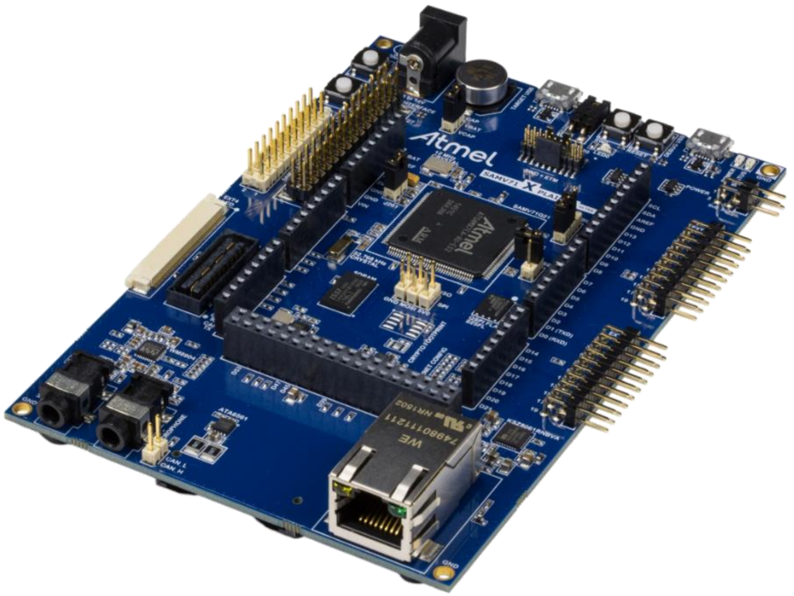


III) Experiments, platform: AURIX Kit TC297

- Development board for the AURIX TriCore TC297 microcontroller
- The TC297 microcontroller targets demanding automotive and industrial applications such as Advanced Driver Assistance Systems, Sensor fusion, Engine management or Transmission control
- Key features:
  - 3x32-bit TriCore CPU running at up to 300 MHz
  - 8MB embedded Flash and 728 Kbytes SRAM
  - Cryptography: TRNG and AES128 encryption

Experimental Platform II: SAM V71 Xplained Ultra

- Evaluates the ATSAMV71Q21 microcontroller (SAM V71 Xplained Ultra Evaluation Kit)
- SAM V series are based on ARM Cortex-M7 processors and are especially focused on the in-vehicle infotainment connectivity for Audio Amplifiers, Telematic Control Units or Head Units
- Key features:
  - 32-bit ARM Cortex –M7 running at up to 300 MHz
  - 2048 Kbytes embedded Flash and 384 Kbytes SRAM
  - Cryptography: TRNG, AES and Hash Algorithms



Experimental Platform III: HTC One (M7)

- Designed and manufactured by HTC
- Released in March 2013
- Equipped with Android 4.1.2
- CPU: 1.7 GHz quad-core Krait 300
- Memory: 2 GB LPDDR2 RAM
- Storage: 32 GB

Cryptographic libraries that we used for testing

- For basic RSA/DSA operations, WolfSSL, a lightweight embedded SSL/TLS library
  https://www.wolfssl.com/

- For BLS signatures, the PBC Library made available by Ben Lynn
  https://crypto.stanford.edu/pbc/

  D. Boneh, B. Lynn, and H. Shacham. *Short signatures from the Weil pairing*, 2004

- The BLS library is also ported on Android by:

  A. De Caro and V. Iovino. *jPBC: Java pairing based cryptography*, ISCC 2011

- For group signatures and identity based encryption, the PBC library by Unterlauger et al.
  https://github.com/IAIK/pairings_in_c

  T. Unterluggauer and E. Wenger. *Efficient pairings and ECC for embedded systems*. Workshop on Cryptographic Hardware and Embedded Systems, 2014

Results for the pairing based library of Unterlauger et al. (Cortex M7)

- Computational time is high for real-time need
- Not really suitable for 100ms communication cycles, e.g., v2v status messages
- May cope with slower processes, e.g., software updates

Platform	Core	Flash size	RAM size	Frequency	Manufacturer
ATSAMV71	Cortex-M7	2MB	384KB	300MHz	Microchip
TC297	TriCore 1.6P	8MB	728KB	300MHz	Infineon
HTC One M7	Krait 300	n/a	n/a	1.7GHz	Qualcomm

Function	Procedure	Flash [bytes]	Runtime [ms]	Signature [bytes]
RSA	MakeRsaKey	34176	7783	128
	RsaSSLSign		273	
	RsaSSLVerify		47	
RSA	MakeRsaKey	34176	54122	256
	RsaSSLSign		1281	
	RsaSSLVerify		155	
DSA	InitDsaKey	24628	38350	40
	DsaSign		155	
	DsaVerify		311	
IBE(BN158)	GenParams	20868	315	N/A
	DerivePrivateKey		190	
	EncapsulateKey		309	
	DecapsulateKey		215	
IBE(BN254)	GenParams	20868	941	N/A
	DerivePrivateKey		603	
	EncapsulateKey		971	
	DecapsulateKey		566	
SGS(BN158)	SgsInit	23416	400	252
	SgsSign		713	
	SgsVerify		1073	
	SgsOpen		68	
SGS(BN254)	SgsInit	23416	1247	396
	SgsSign		2128	
	SgsVerify		3099	
	SgsOpen		231	
HWANG scheme(BN158)	HwangInitParams	27340	474	232
	HwangGenerateUsk		112	
	HwangSign		684	
	HwangVerify		1226	
HWANG scheme(BN254)	HwangInitParams	27340	1571	364
	HwangGenerateUsk		373	
	HwangSign		2058	
	HwangVerify		3518	
Bilinear pairings (BN158)	PbcMapOptAteOptimizedStd	16988	161	N/A
Bilinear pairings (BN254)	PbcMapOptAteOptimizedStd	16988	405	N/A

Results for the pairing library of Unterlauger et al. on Infineon

- Surprisingly, the Infineon core was generally faster than the Cortex M7
- Memory requirements are higher but this may vary due to specific compiler optimizations
- Pairings suitable for key-exchange

Function	Procedure	Flash [bytes]	Runtime [ms]	Signature [bytes]
DSA	DsaSign	61988	29.4	40
	DsaVerify		57.8	
IBE (BN158)	GenParams	29442	78.9	N/A
	DerivePrivateKey		46.8	
	EncapsulateKey		78	
	DecapsulateKey		54.9	
IBE (BN254)	GenParams	30268	225	N/A
	DerivePrivateKey		146	
	EncapsulateKey		235	
	DecapsulateKey		138.4	
SGS (BN158)	SgsInit	30302	81.2	252
	SgsSign		173.6	
	SgsVerify		264.2	
	SgsOpen		15.7	
SGS (BN254)	SgsInit	31234	264.2	396
	SgsSign		511.6	
	SgsVerify		745.6	
	SgsOpen		53.4	
HWANG scheme (BN158)	HwangInitParams	29704	114	232
	HwangGenerateUsk		24.12	
	HwangSign		166.2	
	HwangVerify		304.4	
HWANG scheme (BN254)	HwangInitParams	30778	368.8	364
	HwangGenerateUsk		80.80	
	HwangSign		488	
	HwangVerify		853	
Bilinear pairings (BN158)	PbcMapOptAteOptimizedStd	26504	40.9	N/A
Bilinear pairings (BN254)	PbcMapOptAteOptimizedStd	27320	102.7	N/A

Platform	Core	Flash size	RAM size	Frequency	Manufacturer
ATSAMV71	Cortex-M7	2MB	384KB	300MHz	Microchip
TC297	TriCore 1.6P	8MB	728KB	300MHz	Infineon
HTC One M7	Krait 300	n/a	n/a	1.7GHz	Qualcomm

Results for BLS signatures (Boneh, Lynn, Sacham)

- Generally, the Android device is faster, but variations exists (will be subject to future investigations)
- Pairing-based operations can be easily handled by Android devices

Function	Procedure	Flash [bytes]	Runtime [ms]	Signature size [bytes]
BLS-A	BLSSign BLSVerify	324564	4828 7286	64
BLS-A1	BLSSign BLSVerify	324564	37516 31122	130
BLS-D159	BLSSign BLSVerify	324564	251 5237	20
BLS-D201	BLSSign BLSVerify	324564	488 10459	26
BLS-D224	BLSSign BLSVerify	324564	683 12581	28
BLS-E	BLSSign BLSVerify	324564	41072 36621	128
BLS-F	BLSSign BLSVerify	324564	236 25621	20
BLS-G149	BLSSign BLSVerify	324564	227 18640	19

Platform	Core	Flash size	RAM size	Frequency	Manufacturer
ATSAMV71	Cortex-M7	2MB	384KB	300MHz	Microchip
TC297	TriCore 1.6P	8MB	728KB	300MHz	Infineon
HTC One M7	Krait 300	n/a	n/a	1.7GHz	Qualcomm

Function	Procedure	Flash [bytes]	Runtime [ms]	Signature size [bytes]
BLS-A	GenKeys BLSSign BLSVerify	N/A	351 685 1584	64
BLS-A1	GenKeys BLSSign BLSVerify	N/A	3226 2122 10129	130
BLS-D159	GenKeys BLSSign BLSVerify	N/A	3561 156 10796	20
BLS-D201	GenKeys BLSSign BLSVerify	N/A	4122 252 12815	26
BLS-D224	GenKeys BLSSign BLSVerify	N/A	12019 398 15630	26
BLS-E	GenKeys BLSSign BLSVerify	N/A	642 1707 2972	128
BLS-F	GenKeys BLSSign BLSVerify	N/A	5912 184 86174	20
BLS-G149	GenKeys BLSSign BLSVerify	N/A	8823 195 34168	19

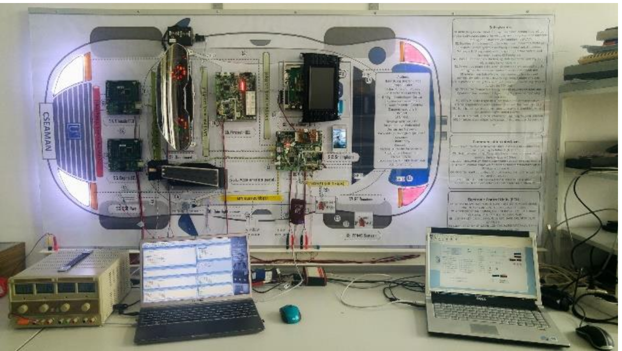
Platform	Core	Flash size	RAM size	Frequency	Manufacturer
ATSAMV71	Cortex-M7	2MB	384KB	300MHz	Microchip
TC297	TriCore 1.6P	8MB	728KB	300MHz	Infineon
HTC One M7	Krait 300	n/a	n/a	1.7GHz	Qualcomm

Conclusions

- Several automotive based scenarios seem practical for pairing-based cryptography: vehicle bus security, v2v communication, software updates, etc.

- Computational costs are high, e.g., from hundred ms up to seconds, but feasible on high-end automotive controllers for certain applications

- Future work:
  - Concrete applications for any of the aforementioned scenarios



http://www.aut.upt.ro/~bgroza/presence.html

http://www.aut.upt.ro/~bgroza/cseaman.html