Development of an AUTOSAR Compliant Cryptographic Library on State-of-the-Art Automotive Grade Controllers

Pal-Ştefan Murvay, Alexandru Matei, Cristina Solomon and Bogdan Groza
Faculty of Automatics and Computers
Politehnica University of Timisoara, Romania
Email: {pal-stefan.murvay, bogdan.groza}@aut.upt.ro, alexandru.matei@msn.com, cristina_solomon@ymail.com

***Abstract*—In the light of the recently reported attacks on intra-vehicle networks, it has become clear that cryptography is vital for assuring the security of in-vehicle communications. The current preoccupation of industry professionals in this direction is proved by the inclusion of a comprehensive cryptographic extension in the recent-most version of the AUTOSAR (AUTomotive Open System ARchitecture) standard. In this work we try to give an answer on how prepared are current state-of-the-art automotive controllers for implementing cryptographic primitives and what is the exact cost of software implementations. We take into account automotive grade controllers that range from some of the most constrained platforms, e.g., from 8051 based tire sensors with 8-bit cores, up to 32-bit Infineon TriCore architectures, as well as devices that lay in between these two. We provide experimental results on several symmetric cryptographic primitives, i.e., block ciphers and hash functions, mainly focusing on the lightest constructions proposed in the literature, e.g., Speck, Katan, Blake, as well as on past or current standards, e.g., AES, SHA2 or SHA3. As expected, the results are sparse, some of the platforms being well prepared, capable to easily handle software implementation or carrying dedicated hardware, while for others no dedicated hardware exists while software implementation of current cryptographic standards cannot be handled, especially with the overhead incurred by the cohesion to the AUTOSAR standard.**

1. Introduction

As cars evolved in a similar manner to modern computers, there are little doubts that cryptography is the only alternative in assuring the necessary security objectives for inter and intra vehicular networks. With few exceptions, all of the attacks that are reported so far, e.g., [9], [12], [13] take advantage of the absence of cryptographic security (message authentication in particular) on the in-vehicle communication interfaces.

The automotive industry was quite determined in designing in-vehicle buses and currently there are several communication buses that compete, persist or can be simultaneously found in a single vehicle. The traditional Controller Area Network (CAN) is the most wide spread in-vehicle bus, it also has a recently updated version, the CAN-FD (CAN with Flexible Data Rate) that can accommodate higher data rates and message length. The low cost Local InterConnect (LIN) is generally used for connecting peripherals, e.g., doors, windows, etc. Recently designed, high performance buses such as FlexRay or BroadR-Reach (an Ethernet based bus) can be found in high-end vehicles. But regardless of the communication layer, the same invariant persists: there is no security, except for standard CRC codes that are required for the correctness of the transmission. While bandwidth is a realistic drawback in implementing security for CAN or LIN, for the newer communication layers, e.g., CAN-FD, FlexRay and BroadR-Reach, adding security is clearly possible - if the suitable cryptographic primitives exists. Here we try to answer to this question by determining to what extent current automotive grade controllers are well prepared for handling cryptography.

Given the intrinsic challenges in designing and building vehicles components that rely on software and (inevitably) originate from distinct manufacturers, the automotive industry was also opened toward standardization. The AUTOSAR (AUTomotive Open System ARchitecture) initiative, started in 2003, has the objective of standardizing software architectures. Recently, AUTOSAR started to include specifications for cryptographic primitives. The AUTOSAR CAL (Specification of Crypto Abstraction Library) [3] and CSM (Specification of Crypto Service Manager) [4] provide basic cryptographic functionalities for software applications. CAL is designed as an independent library that relies on software implementations. CSM is designed as a part of the system services and to be accessed by the application through the run-time environment, it can rely on software or hardware implementations. Both CAL and CSM provide specifications for the following standard cryptographic primitives:

(i) Hash functions, which can be used for integrity checks,
(ii) MAC (Message Authentication Codes), which are used for checking authenticity,
(iii) Symmetric encryptions, in particular via block-ciphers, which are mainly used for protecting information against eavesdroppers,
(iv) Asymmetric encryptions, i.e., public-key encryptions, having the main utility in exchanging symmetric session keys,
(v) Digital signatures, which are needed to test authen-

ticity of messages based on a public (non-secret) key.

In addition to these, there are a number of building blocks that are required for key management. Randomness is also addressed as it is needed for generating fresh cryptographic keys. Key generation, key derivation, key exchange and wrapping interfaces are also specified by the standard.

Due to intrinsic limitations in terms of computational power and storage space on the devices that we work here, as well as due to the more limited scope of our work, here we focus on symmetric cryptographic techniques alone. We do target two classes of symmetric primitives: block ciphers and hash functions. While Message Authentication Codes (MACs) are of relevance they are not addressed since these are built on the previous two primitives, e.g., the CBC-MAC or the HMAC, and the computational results can be easily derived. AUTOSAR does not specify which primitives should be used for each type of service leaving the task of choosing the appropriate algorithms to the system design phase. We now enumerate our choices for block ciphers as well as hash functions along with the motivation behind our choice:

(i) AES - the Advanced Encryption Standard is the current standard in symmetric cryptography [14] and neglecting it is not an option for any realistic deployment, it uses block of 128 bits and keys of 128, 192 or 256 bits,
(ii) SPECK - is a recent proposal from NSA [5] and so far this is the lightest block cipher available for software implementations, block size is of 2 words of 16, 24, 32, 48 or 64 bits while the key is 2, 3 or 4 words, we choose it as a baseline for performance,
(iii) PRESENT - is another lightweight block cipher with 64 bit blocks and keys of 80 or 128 bits [7], we choose it since it proved to be the maximum that can be handled by our most constrained platform, i.e., an 8051 based TPM sensor,
(iv) KATAN - is a lightweight block cipher with blocks of 32, 48 or 64 bits and keys of 80 bits [11], we choose it for being a popular choice in many research works.
(v) MD5 - while insecure and recommended for modern applications, it is one of the lightest designs for hash functions and we use it a baseline in performance, it outputs digests of 128 bits,
(vi) SHA1 - is also considered insecure for today needs, but it is lighter than the standard SHA2 which is not suitable for many of our platforms,
(vii) SHA2 - was the standard cryptographic hash until 2015, it is quite demanding from a computational point of view, it outputs digests on 256, 384 or 512 bits,
(viii) SHA3 - is the standard released by NIST on 2015, it is based on Keccak [6], the winner of the SHA-3 competition, it outputs digests of 224, 256, 384 or 512 bits,
(ix) Blake2 - is based on Blake, one of the SHA3 finalists [1], and we choose it for being the lightest modern design, it outputs digests of 224, 256, 384 or 512 bits.

TABLE 1. TOP 10 AUTOMOTIVE SUPPLIERS ACCORDING TO [10]

Rank	Company	Market Share
1	Renesas Electronics Corporation	10,4%
2	Infineon Technologies	9,3%
3	STMicroelectronics	7,4%
4	Freescale Semiconductor	7,2%
5	NXP	6,4%
6	Robert Bosch	5,6%
7	Texas Instruments	5,5%
8	On Semiconductor	3,7%
9	Toshiba	2,5%
10	Micron Technology	2,4%

2. Target devices

For comprehensive evaluation of the chosen cryptographic primitives, we made a mixed selection of automotive grade platforms (on 8, 16 and 32-bit cores). While not exhaustive, this selection of automotive grade microcontrollers was made with the intent of covering a wide range of ECUs from all in-vehicle domains and market areas (i.e. devices ranging from the low-end to the high-end sector). Since the architecture of the core is the prime factor that influences the performance of code across platforms we mainly selected one representative of each microcontroller family included in our study. The sole exceptions are the TriCore and RH850 platforms for which we selected two representatives to illustrate performance similarity. The manufacturer was also taken into consideration as we picked only microcontrollers from the top suppliers of the automotive industry [10] which are presented in Table 1.

Our target devices and the microcontroller families of which they are part of are presented in what follows while Table 2 summarises this presentation.

(i) ***Freescale S08 (8bit).*** The *S08* family from Freescale[1] is an 8-bit platform with several members designed for various automotive applications such as HVAC (Heating Ventilation and Air Conditioning), lighting, doors, window lift, seat control, instrument cluster and airbags. S08 family members can offer between 2 and 128KB of Flash, up to 8KB of RAM and maximum operating frequencies of 8, 20 and 40MHz. We selected the S08AC128 as the representative of the S08 family for our tests. It comes with the top options of the family: 128KB Flash, 8KB RAM and 40MHz operating frequency.
(ii) ***Infineon SP37 (8bit, 8051 based).*** On the 8-bit architecture side we also looked at devices designed for very specific applications. Such is the case of the Infineon *SP37* tire pressure monitoring sensor (TPMS) which is built around an 8051 compatible 8-bit microcontroller. The SP37 is a very constrained platform, having 6 KB Flash memory and only 256 bytes of RAM of which the upper 63 bytes are used by the TPMS specific ROM functions library. The

1. Freescale has been recently merged under the NXP name, we will refer to Freescale as the manufacturer for the devices known under this brand

flash memory also has further restrictions as it allows only 2 KB for the user program.

(iii) ***Freescale S12 (16bit).*** Our first choice for 16-bit devices is the Freescale *S12* and *S12X* family which covers a broad array of mid-range vehicle body applications. The S12X range extends the S12 core with the integration of the XGATE coprocessor to bring higher performances. Up to 1M of Flash and 64KB of RAM are available with the S12(X) family members that operate at 16 to 80MHz. We selected S12XDT512 which can operate at 80MHz and is equipped with 512KB of Flash and 20KB of RAM.

(iv) ***Freescale S12Z (16bit).*** Built on the S12 technology, the *S12Z* 16-bit microcontrollers are employed for implementing entry level instrument clusters or sensors and actuators for body, chassis and safety. S12Z devices come with 16-192KB Flash, 1–12KB RAM and operating frequencies up to 64MHz. For our tests we used the S12ZVH64 derivative which has 64KB Flash, 4KB RAM and runs at 64MHz.

(v) ***Renesas RL78/D1x (16bit).*** The *RL78/D1x* is a 16-bit low power microcontroller produced by Renesas for low-end instrument clusters. Members of this group operate at 32MHz and provide 24–512KB of Flash and 2–24KB of RAM. The RL78/D1A, equipped with 512KB of Flash and 24KB of RAM was included in our study.

(vi) ***Texas Instruments MSP430 (16bit).*** Another 16-bit platform on our list is *MSP430* from Texas Instruments. The MSP430 family includes several members that target various body and infotainment applications offering up to 120KB of Flash, 8KB RAM and 16MHz operating frequency. We used an MSP430F2274, with 32KB Flash and 1KB RAM, as a representative of this family.

(vii) ***Freescale Qorivva MPC56xx (32bit).*** We switch to 32-bit architectures with the Freescale Qorivva *MPC56xx* family designed for applications in engine management, powertrain, ADAS (Advanced Driver Assistance Systems), BCM (Body Control Module), gateways, chassis and safety and instrument clusters from low up to high-end projects. This family is equipped with the e200 Power Architecture core and can operate at 32-270MHz. Some family members have two e200 cores while the MPC564xB-C has an e200z4 and comes with an integrated security module that offers functionalities for generating random numbers and using AES-based encryption and authentication. Given the wide range of target applications the memory options in this family also has a considerable coverage with up to 6000KB Flash and 1088KB RAM. As a member of this family, the MPC5606B, which comes with 1MB Flash, 80KB RAM and a top operating frequency of 64MHz, was employed in our tests.

(viii) ***Freescale iMX6 (32bit, ARM based).*** Another 32-bit platform from the same manufacturer is the *iMX* application processor family. The I.MX6DualLite is a 32-bit application processor that features two ARM Cortex-A9 cores which are operating at speeds of up to 800MHz. In terms of memory is has an 128KB RAM and only 96KB of flash destined for bootloader functionality as this architecture mainly relies on external flash. This processor is designed to be used in applications such as: automotive navigation and entertainment, graphics rendering for Human Machine Interfaces (HMI), high-performance speech processing with large databases, audio playback, video processing and display. It also features hardware enabled security functionalities that can be used in e-commerce, digital rights management, information encryption, secure boot and secure software downloads.

(ix) ***Infineon Tricore TC1797 & TC1782 (32bit).*** Infineon's solution for computationaly demanding applications is the *TriCore* architecture. The AUDO family offers a variety of microcontrollers with 1-4MB of Flash, 48-288KB RAM and operating frequencies between 80 and 300MHz for mid- to high-end powertrain, chasis and safety applications. We considered 2 members of this family, TC1797 and TC1782, each from a different subcategory, AUDO Future and AUDO MAX respectively. Both are based on the same core version (TriCore V1.3.1), can work at 180MHz and have 176KB of RAM, therefore we expected to obtain similar performance results. TC1797 has 4MB of Flash , while TC1782 comes with 2.5MB of Flash. As with most of the members of the AUDO family, our 2 choices come with an additional coprocessor called PCP (Peripheral Control Processor) designed to manage on-chip module decreasing the load of the main core.

(x) ***Renesas RH850 (32bit).*** Our last selections come from the Renesas *RH850* 32-bit family which offers a wide array of single and multi core devices intended for virtually all in-vehicle functionalities of the high-end sector: powertrain, instrument cluster, body, safety, ADAS and other. The cores of the RH850 family members come in two flavours, G3M for improved data processing and G3K with a simplified design and improved low power behaviour. Given the variety of target applications the options available in this family are also very diversified: up to 8MB Flash, 512KB RAM and maximum, frequencies of 320MHz. We used two members of this family to represent each of the two core architectures available. The first one is RH850/F1L, equipped with a G3K core running at 80MHz, 2MB of Flash and 192KB of RAM. RH850/E1x-FCC1, the second one, is powered by a G3M core running at 320MHz with 4MB of Flash and 352KB of RAM.

3. Implementation details

We first give a brief overview on the general structure of an AUTOSAR compliant deployment. Then we give concrete details on our specific implementations.

TABLE 2. PLATFORMS TARGETED IN OUR WORK

Device	Core	Flash size	RAM size	Frequency	Manufacturer
S08AC128	S08	128KB	8KB	40MHz	NXP(Freescale)
SP37	8051	6B	256B	12MHz	Infineon
S16XDT512	S12(X)	512KB	20KB	80MHz	NXP(Freescale)
S16ZVH64	S12Z	64KB	4KB	64MHz	NXP(Freescale)
RL78/D1A	RL78	512KB	24KB	32MHz	Renesas
MSP430F2274	MSP430	32KB	1KB	16MHz	Texas Instruments
MPC5606B	e200	1MB	80KB	64MHz	NXP(Freescale)
iMX6	Cortex-A9	96KB	128KB	800MHz	NXP(Freescale)
TC1782	TriCore 1.3.1	2.5MB	176KB	180MHz	Infineon
TC1797	TriCore 1.3.1	4MB	176KB	180MHz	Infineon
RH850/F1L	RH850 G3K	2MB	192KB	80MHz	Renesas
RH850/E1x-FCC1	RH850 M3K	4MB	352KB	320MHz	Renesas

3.1. AUTOSAR generic system structure

According to the specification [2] an AUTOSAR compliant system follows a layered software architecture comprised of three main software layers: Application, Runtime Environment (RTE) and Basic Software (BSW). The BSW layer contains drivers and services needed by the application. The RTE is an interfacing layer providing application access to BSW functionality. An additional Libraries layer (LIB) provides a container for various functions that are needed by system modules. LIB modules can be accessed by all other AUTOSAR layers but can only call functions residing within LIB. Figure 1 presents a simplified representation of the AUTOSAR layer structure and the positioning of the CAL and CSM modules in this architecture.

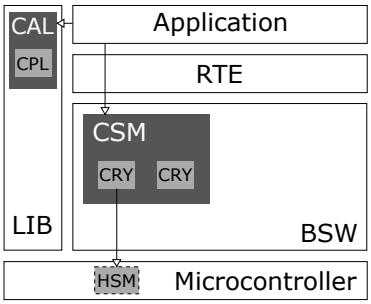


Figure 1. AUTOSAR layered architecture

Both CAL and CSM consist of a wrapper layer acting as the interface for the calling modules to the primitive implementation layers. The implementation layer of CAL is called CPL (Cryptographic Primitive Library) while the corresponding layer of CSM is named CRY (Cryptographic Library Module). CPL and CRY modules can directly provide the implementation of a certain primitive or an interface to a cryptographic library component. The second alternative is particularly required by the AUTOSAR specification when both CRY and CPL use the same building blocks for avoiding duplicate code. In our particular case both CAL and CSM provide their services by calling the same set of primitive implementations through corresponding CPL and CRY interfaces. Obviously, it is hard to find a practical scenario which would require the access to the same software implementations of cryptographic primitives through 2 types of interfaces but keeping with the configurable nature of AUTOSAR this allows the system designers to chose between using CAL and CSM.

While in the case of software implementations there are few differences between using CAL and CSM, a significant distinction comes in that the CSM can use cryptographic HW if available on the microcontroller. The CRY accesses the functionality of the HSM (Hardware Security Module) through a dedicated driver. From the upper layers perspective, choosing between a software implementation and one based on an HSM is the same as selecting between different software implemented algorithms for the same type of cryptographic service and only requires the calling of the wrapper functions with the corresponding configuration identifier.

Most of the cryptographic primitives which can be accessed through CAL and CSM are implemented for usage in streaming mode (hash functions and block ciphers fit in this category). This means that their interface consists of:

- a Start function used for initialising the algorithm context,
- an Update function which is called repeatedly to process smaller blocks of data from a larger segment,
- a Finish function which is called to finalise the processing before getting the result.

Some services (e.g. random number generation) are deviating from this rule and are to be used through a single function call.

3.2. Specifics of primitive implementations

The implementations for the cryptographic primitives included in our library are based on the reference source codes (in the case of hash functions) and on the open-source BLOC library [8] (in the case of block ciphers). For the moment, it was not in our focus to make platform dependent optimizations for these implementations. Mostly we had to adapt code due to specific compiler needs (e.g. some compilers cannot handle 64-bit variables). For SHA2 and SHA3 we only implemented their 256 bit block

versions while for Blake2 we used the Blake2s implementation, which is optimised for 8 to 32-bit platforms, with the same digest size of 256 bit. For block ciphers, we used AES with 128-bit key, the bit-sliced implementation with 32 slices of Katan32, Present for both 80 and 128-bit keys and Speck with 128-bit block and key.

From our list of target devices only the iMX6 platform provides hardware support for implementing cryptography. We therefore made use of the hardware based AES implementation by using the Cryptographic Acceleration and Assurance Module (CAAM). CAAM can be programmed through a ring interface which works as follows: the processing requests are given as entries in the input ring while results are taken from the output ring. This mechanism allows multiple jobs to be queued for hardware execution enabling the CPU to execute other tasks until the CAAM signals the end of each job by means of dedicated flags.

In the case of the SP37 we could not use all of the library implementations due to the major memory constraints. However, we tried to make target-specific implementations considering also the device basic scope and functionality. In contrast to the other platforms used in this paper, this device is designed for a specific application - tire sensors. The AUTOSAR architecture was not used for this platform due to the aforementioned constraints. From the selected hash functions we attempted to fit an MD5 implementation on the chip as it is on the lightweight side. In spite of our efforts of manually optimizing the code for size and the additional compiler optimizations we were unable to fit MD5 in the SP37 flash memory. Our best implementation still exceeded flash capacity by 290 bytes (14%). Similar unsuccessful attempts were made for implementing the other hash functions in our study. For block ciphers we were able to use the same implementation of Present as on the other platforms. For Speck platform specific optimizations were needed to fit an implementations with a block size larger than 32 bits, this allowed us to fit Speck with a 64 bit block and 128 bit keys.

4. Experimental results

We used our AUTOSAR CAL implementation to test the capabilities of each target platform. Only the SP37 had to be treated separately as we could only test it with specific implementations and not with the generic AUTOSAR compliant source code.

We focused on two performance metrics which are important in the development of embedded systems in general and for the automotive grade software in particular: execution speed and memory consumption. Execution speed is important as it can affect the speed at which one device can send messages containing authentication data or cipher texts. The flash memory available on a certain device may not be enough to hold the application and the cryptographic library, therefore it is essential to have an image on the memory requirements for using such a library.

4.1. Execution speed

For the majority of platforms, the code was deployed and executed on the platform. In the case of the S08, RL78 and RH850 platforms a simulator was used for evaluating execution speeds. For fairness in comparison, basic compiler settings were used in all cases without enabling specific optimisations for speed or size. For the block ciphers, besides Katan which had 80 bits keys for all variants, we used 128-bit keys for all input sizes.

Measurement for execution speed was done using an oscilloscope for the platforms for which we had the corresponding hardware. For the other platforms, simulator capabilities for clock cycle counting were used. Table 3 shows the execution speed of all primitives in our CAL implementation on the selected target platforms for various input sizes. For the block ciphers, the values represent the time for encryption of the input block. The approximation for very long inputs was calculated as the difference in cycles/bytes between the result for 4096 byte inputs and 2048 byte inputs divided by 2048.

It can be noticed that the ranking of the cryptographic primitives may differ depending on the platform. This is caused by the fact that the same code is used on all platforms without any specific improvements. Therefore, the implementation may prove to be better suited on some platforms than the others. One clear example for this is Speck which will outperform the other ciphers in most cases but proves to be slower on S08 and S12 devices.

Current uses of cryptographic primitives in automotive applications are mainly for device reprogramming functionalities the where the amount of data which needs to be processed is very large divided into hundreds or thousands of messages (depending on the size of the firmware that needs to be signed). We therefore graphically illustrate the execution times for 4096 byte inputs in Figures 2 and 3. These plots were synthetically obtained based on the determined cycle/byte value and considering that the frequency used on each platform is the maximum possible (Table 2).

Hardware support for AES on iMX6. From our list of selected devices, only iMx6 has hardware support for cryptographic primities, i.e., AES. We also evaluated the AES HW-based implementation on iMx6 which for 64-byte messages was only 8.9 times faster that the SW implementation - the longer than expected execution time is due various calls from the software layer and measurement difficulties due to the multi-tasking OS that runs on the iMx6. For the longer 4KByte messages the execution time was 67.5 faster and at 379.33 cycles/byte is faster than the software implementation in any of our target platforms.

Limitations on MSP430 and SP37. While testing the implementations for result correctness we found that the SHA3 implementation was not providing correct results on our MSP430 device due to stack overflowing during execution. We could not resolve this issue by any further straightforward optimisation; for this reason the results in Table 3 for SHA3 on MSP430 are not provided. On the SP37 we only evaluated Present and Speck as these were the only primitives we could implement on the device.

TABLE 3. EXECUTION SPEED (CYCLES/BYTE) FOR THE IMPLEMENTED PRIMITIVES ACROSS VARIOUS PLATFORMS

Platform	Input size	Cryptographic primitive (block size and key length)								
		MD5 128	SHA1 160	SHA2 256	SHA3 256	Blake2 256	AES 128-128	Katan 32-80	Present 64-128	Speck 128-128
S08	8	34177.88	60201.75	135030.38	1952959.50	81942.25	7481.75	356709.00	94472.63	64011.63
	64	8367.11	15421.53	33319.19	253866.42	10255.02	2688.56	58850.25	85394.72	20599.17
	576	4573.71	8796.18	18295.47	128830.82	9784.00	2377.86	39572.49	84241.97	17213.73
	1536	4277.35	8278.58	17121.74	114808.04	9748.71	2353.54	37437.09	84151.91	16949.87
	4096	4166.21	8084.48	16681.59	110612.63	9735.48	2344.72	37265.64	84118.14	16518.22
	long msgs	4099.53	7968.02	16417.50	106819.63	9727.54	2349.27	37162.77	84097.88	16126.53
S12	8	5052.38	14418.13	31543.63	445194.00	13409.00	3821.13	56714.88	33277.13	9054.75
	64	1205.66	3679.77	7756.36	58188.17	1683.81	1495.81	10254.45	31001.03	2943.23
	576	645.36	2092.81	4247.22	28985.94	1584.79	1373.15	7745.72	30712.00	2463.86
	1536	601.59	1968.65	3973.07	25769.49	1577.28	1363.56	7464.92	30754.53	2426.41
	4096	585.18	1922.15	3870.26	28391.24	1574.46	1359.97	7446.62	30680.96	2412.36
	long msgs	575.33	1894.25	3808.58	31121.57	1572.77	1357.81	7432.13	30675.88	2403.94
S12Z	8	2076.00	8016.00	12864.00	428000.00	9424.00	9824.00	109760.00	44720.00	5936.00
	64	385.50	2070.00	2940.00	54900.00	1204.00	4075.00	16260.00	40050.00	1656.00
	576	156.89	1195.56	1522.22	27888.89	934.44	3827.78	9600.00	39500.00	1288.89
	1536	138.75	1127.08	1412.50	24791.67	914.58	3808.33	8854.17	39416.67	1260.42
	4096	132.03	1101.56	1370.31	23875.00	906.25	3796.88	8812.50	39453.13	1250.00
	long msgs	127.97	1087.50	1345.31	23031.25	901.56	3787.50	8781.25	39468.75	1243.75
RL78 D1A	8	849.75	12274.00	4613.50	461547.87	2876.75	2909.62	40351.12	19021.87	1307.62
	64	182.23	3075.38	1099.17	60454.98	362.22	1063.52	7464.80	17963.34	395.55
	576	87.93	1714.38	587.06	29950.12	284.22	947.78	5985.94	17828.93	323.95
	1536	80.56	1608.05	547.05	26609.24	278.27	938.74	5828.97	17818.43	318.36
	4096	77.80	1568.18	532.05	25595.80	276.04	935.35	5811.54	17814.49	316.26
	long msgs	76.15	1544.26	523.06	24700.48	274.70	933.31	5801.09	17812.13	315.00
TC1782	8	281.25	1102.50	1327.50	117225.00	1399.50	1287.00	10687.50	7863.75	632.25
	64	41.23	271.69	290.81	14821.88	168.75	502.31	1614.38	7228.13	168.47
	576	16.47	151.88	149.84	8015.63	129.69	465.00	962.50	7109.38	128.13
	1536	14.58	142.50	138.75	7183.59	126.80	461.13	888.28	7101.56	124.92
	4096	13.84	139.09	134.91	6952.15	125.68	459.67	883.30	7110.35	123.71
	long msgs	13.43	137.20	132.36	6723.63	124.80	458.79	880.66	7110.35	123.13
TC1797	8	282.60	1113.75	1332.00	117225.00	1401.75	1284.75	10676.25	7751.25	627.75
	64	41.23	271.69	291.38	14821.88	169.31	501.75	1611.56	7059.38	167.91
	576	16.50	151.88	150.63	8015.63	129.84	464.38	962.50	6984.38	127.97
	1536	14.55	142.50	139.69	7183.59	126.80	461.13	888.28	6972.66	124.69
	4096	13.84	138.87	135.57	6952.15	125.68	459.67	883.30	6978.52	123.66
	long msgs	13.41	136.76	133.15	6723.63	124.98	458.79	880.66	6978.52	123.05
MSP430	8	854.13	7284.13	7381.50	N/A	4525.00	3310.25	43154.63	20993.88	3154.63
	64	177.27	1825.91	1779.19	N/A	567.38	1054.02	9180.23	19613.02	993.30
	576	82.92	1020.67	958.44	N/A	467.96	875.49	6706.38	19309.26	824.68
	1536	75.55	957.76	894.32	N/A	460.36	861.64	6426.41	19288.99	811.27
	4096	72.78	934.17	870.27	N/A	457.51	856.39	6408.12	19277.00	806.07
	long msgs	71.13	920.02	855.84	N/A	455.80	853.21	6397.15	19270.41	802.95
MPC5606B	8	403.00	2280.00	2408.00	158800.00	1772.00	2976.00	43154.63	16960.00	844.00
	64	74.13	578.75	563.75	19125.00	224.00	1277.50	4405.00	15850.00	231.75
	576	29.72	329.44	297.78	10041.67	176.11	1215.28	2605.56	15777.78	178.06
	1536	26.25	309.90	277.08	8979.17	172.50	1210.42	2395.83	15750.00	173.96
	4096	24.96	302.73	269.14	8671.88	171.09	1210.94	2390.63	15742.19	172.46
	long msgs	24.18	298.44	264.45	8390.63	170.31	1209.38	2386.72	15742.19	171.48
iMX6	8	1038.51	4582.12	3861.89	276594.12	5228.98	16213.63	124730.10	113939.10	2804.67
	64	201.24	1107.10	883.64	35730.34	660.64	6084.42	81259.20	110448.11	766.11
	576	81.90	605.34	451.47	18310.13	507.73	5488.07	78579.88	110119.63	583.15
	1536	72.88	565.34	417.84	16308.63	493.68	5441.57	78133.33	110055.31	569.43
	4096	69.38	673.24	405.45	15738.47	488.39	5421.07	78013.76	109997.74	563.99
	long msgs	67.27	718.38	397.84	15196.68	485.57	5407.57	77871.30	109989.03	560.22
RH850 G3K	8	295.48	1858.60	1429.36	129263.00	1576.72	2306.24	16128.72	13551.48	559.48
	64	48.78	469.75	321.59	16505.17	201.09	962.78	2507.11	12593.59	159.78
	576	16.34	266.11	163.54	8673.26	145.95	906.92	1531.55	12471.95	127.97
	1536	13.80	250.20	151.19	7757.24	141.71	902.56	1419.99	12462.45	125.49
	4096	12.85	244.24	146.56	7489.74	140.13	900.91	1413.50	12458.89	124.56
	long msgs	12.28	240.66	143.78	7238.03	139.17	899.91	1409.61	12456.75	124.00
RH850 G3M	8	240.24	1134.48	857.60	101158.48	1205.24	1556.72	11818.84	9340.84	400.84
	64	37.72	284.75	187.72	12867.73	150.95	630.70	1732.48	8522.08	114.58
	576	13.71	163.00	95.90	6824.79	122.26	591.69	1025.05	8419.67	93.10
	1536	11.83	153.49	88.74	6110.75	120.04	588.64	943.02	8411.67	91.44
	4096	11.13	149.92	86.05	5903.47	119.23	587.50	939.02	8408.67	90.81
	long msgs	10.70	147.78	84.44	5706.54	118.73	586.81	936.63	8406.88	90.44

Given the nature of this device, the data that it has to send over a wireless channel will fit be lower than 8 bytes. Therefore, we evaluated them for small input sizes. For an 8 byte input, a Present encryption with an 80 bit key will be executed in 303.3ms (at 16451 cycles/byte) while for Speck with 64 bit block and 128 bit key it will take 29.1ms (at 1578 cycles/byte). We consider these speeds as being reasonable due to the low transmission frequencies required from this kind of devices.

4.2. Memory consumption

To get an image on the impact of the usage of security in AUTOSAR on the flash memory consumption we obtained object code dimensions from map files generated by the linker of each platform. Table 4 shows the flash memory needed to store the primitive implementation while Table 5 illustrates the overhead brought by using the AUTOSAR wrapper.

Adding software security mechanisms to an embedded system should not obstruct the implementation of the main functionalities due to insufficient memory. More than this, usually a memory area is reserved for future use (e.g. updates or hotfixes). With this in mind we evaluated the impact of using security primitives and the AUTOSAR CAL interface on memory usage for our set of target devices and illustrated this in Tables 4 and 5 as the percentage of the total available memory occupied by each of them. In some cases more than 10% of the Flash memory is occupied by the object code of the primitive alone. These occurrences, highlighted in gray, add a considerable limit to the memory that can be used to implement the actual ECU functionality and could be viewed as unacceptable depending on the application complexity. We consider a second category of memory occupation where the occupied percent is smaller than 10% and greater than 5%, this is acceptable when the complexity of the target functionality for the device is moderate. Most of the primitives occupy less than 5% on the majority of platforms which should fit most applications.

Although some of the iMX6 implementations also fall in the first two categories this is perfectly acceptable as the on-chip flash is destined for storing the bootloader which needs considerable less memory than the application. This platform is designed to work with external flash chips that hold the application code. The HW implementation of AES occupies 4874 bytes of flash being $\approx 10\%$ smaller than the software one.

The overhead brought by the usage of the AUTOSAR interface is mostly within reasonable bounds. Exceptions come from the devices that exhibited larger memory consumption for the primitive implementations, namely the S12Z and MSP430 device.

On the SP37 our Present implementation for 80 bit keys occupies 1159 bytes (18.8 % of the flash memory) while the Speck-64-128 implementation takes 1059 bytes (17.2 % of the flash memory). In both cases a considerable part of the available memory is used but as shown in [15] this would still allow the implementation of the basic TPMS functionality along with an authentication protocol.

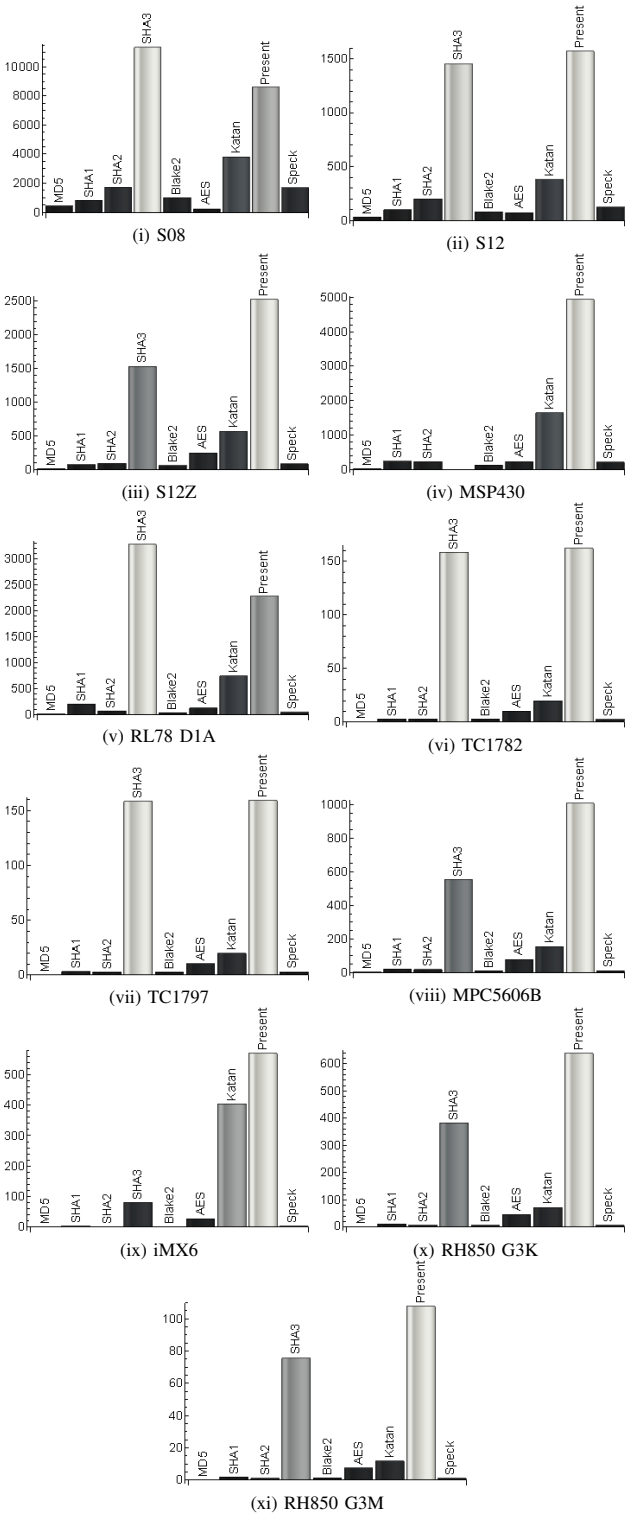
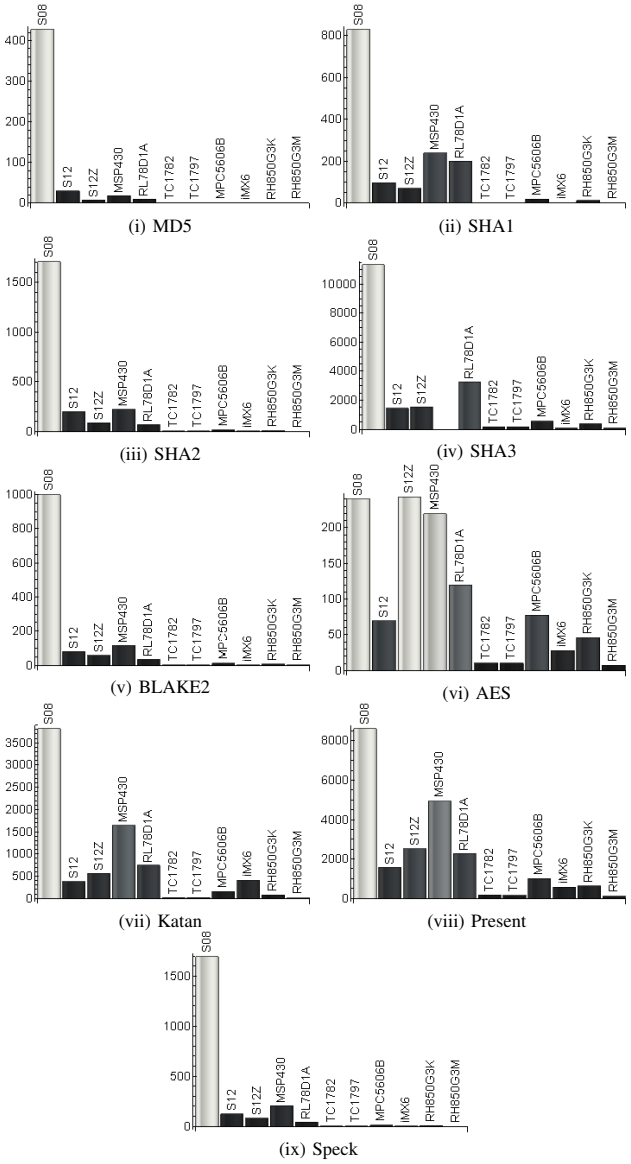


Figure 2. Execution time for 4096 byte inputs(ms) platform-based view

These results are just a baseline and in some scenarios the memory consumption could be lowered through memory code optimizations or could be even increased if by adding speed optimizations the code size increases. We also stress that even though the memory consumption will

TABLE 4. FLASH MEMORY CONSUMPTION OF PRIMITIVE IMPLEMENTATIONS

Platform	Code size																	
	MD5		SHA1		SHA2 256		SHA3 256		Blake2		AES 128-128		Katan 32-80		Present 64-128		Speck 128-128	
	bytes	%	bytes	%	bytes	%	bytes	%	bytes	%	bytes	%	bytes	%	bytes	%	bytes	%
S08	14120	11.03	1227	0.96	2675	2.09	7005	5.47	5275	4.12	2119	1.66	3185	2.49	4113	3.21	4918	3.84
S12	5528	1.08	1042	0.20	2251	0.44	4021	0.79	3455	0.67	1692	0.33	2741	0.54	2186	0.43	2596	0.51
S12Z	5374	8.40	902	1.41	2081	3.25	4922	7.69	4828	7.54	2688	4.20	2761	4.31	4252	6.64	3659	5.72
MSP430	6394	19.98	1338	4.18	2610	8.16	5384	16.83	4046	12.64	1810	5.66	2628	8.21	1776	5.55	1376	4.30
RL78 D1A	8606	1.68	1137	0.22	2304	0.45	5436	1.06	3606	0.70	2611	0.51	3005	0.59	2247	0.44	1309	0.26
TC1782	2856	0.11	730	0.03	1360	0.05	3924	0.16	2634	0.11	1682	0.07	2004	0.08	3080	0.12	1396	0.06
TC1797	2856	0.07	730	0.02	1360	0.03	3924	0.10	2634	0.07	1682	0.04	2004	0.05	3080	0.08	1396	0.03
MPC5606B	3998	0.40	880	0.09	1730	0.17	5916	0.59	3244	0.32	2258	0.23	2518	0.25	3720	0.37	3324	0.33
iMX6	7688	8.01	1516	1.58	2820	2.94	11296	11.77	4628	4.82	5428	5.65	3552	3.70	7756	8.08	5040	5.25
RH850 G3K	2216	0.11	734	0.04	1194	0.06	4464	0.22	2082	0.10	2528	0.13	2182	0.11	2842	0.14	746	0.04
RH850 G3M	2216	0.06	734	0.02	1194	0.03	4464	0.11	2082	0.05	2528	0.06	2182	0.05	2842	0.07	746	0.02



(i) MD5 (ii) SHA1 (iii) SHA2 (iv) SHA3 (v) BLAKE2 (vi) AES (vii) Katan (viii) Present (ix) Speck

Figure 3. Execution time for 4096 byte inputs(ms) primitive-based view

TABLE 5. AUTOSAR INTERFACE MEMORY OVERHEAD

Platform	Hash functions		Block ciphers	
	bytes	%	bytes	%
S08	345	0.27	746	0.58
S12	259	0.05	566	0.11
S12Z	305	0.48	656	1.03
MSP430	360	1.13	480	1.50
RL78 D1A	242	0.05	512	0.10
TC1782	266	0.01	528	0.02
TC1797	266	0.01	528	0.01
MPC5606B	444	0.04	918	0.09
iMX6	711	0.74	1498	1.56
RH850 G3K	288	0.01	624	0.03
RH850 G3M	288	0.01	624	0.02

be the same on all members of the microcontroller families we studied, the memory occupation percentage will differ depending on the particular memory characteristics of each family member. A good illustration of this statement is given by our TriCore and RH850 devices.

5. Conclusion

To the best of our knowledge, our work is the first academic research effort for implementing an AUTOSAR compliant cryptographic library. The main scope was to establish the performance, in terms of execution speed and memory requirements, on a representative set of automotive grade platforms. The results that we obtained are mixed. In general, cryptography is well handled by all platforms, but clear exception exists. For example the SP37 sensor is unable to cope with most of the cryptographic primitives and even the overhead induced by the AUTOSAR interface exceeds its memory. For other platforms, e.g., the case of SHA3 on the MSP430, the code and variables may apparently fit in memory giving the feeling that it can be handled by the controller but the final results are erroneous due to more subtle overflows. In particular the newer SHA3 standard was a poor performer, giving a general feeling that a light-weight hash standard for embedded devices may be needed. The AUTOSAR wrapper brings an overhead which might be considered too large on some devices. In general, given the constrained

nature of embedded devices we expect that AUTOSAR compliant architectures will be present only on high-end devices.

A general impression is that cryptographic hardware is added only to devices that are already from the high-end side of the table, and are capable of good performance for software implementations, while more constrained devices tend to be left behind. It remains an open question on how to secure an entire car body when certain components cannot be secured as they cannot handle cryptography, but this question is out of scope for our work.

References

[1] J.-P. Aumasson, S. Neves, Z. Wilcox-OâĂŹHearn, and C. Winnerlein. Blake2: simpler, smaller, fast as md5. In *Applied Cryptography and Network Security*, pages 119–135. Springer, 2013.

[2] AUTOSAR. *Layered Software Architecture*, 4.2.2 edition, 2015.

[3] AUTOSAR. *Specification of Crypto Abstraction Library*, 4.2.2 edition, 2015.

[4] AUTOSAR. *Specification of Crypto Service Manager*, 4.2.2 edition, 2015.

[5] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, page 175. ACM, 2015.

[6] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3:30, 2009.

[7] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. *PRESENT: An ultra-lightweight block cipher*. Springer, 2007.

[8] M. Cazorla, K. Marquet, and M. Minier. Survey and benchmark of lightweight block ciphers for wireless sensor networks. In P. Samarati, editor, *SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography, Reykjavík, Iceland, 29-31 July, 2013*, pages 543–548. SciTePress, 2013.

[9] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco, 2011.

[10] M. Culver. 2014 Automotive Semiconductors Supplier Rankings Adjusted Based on Further Analysis, IHS Says. http://press.ihs.com/press-release/automotive/2014-automotive-semiconductors-supplier-rankings-adjusted-based-further-ana. 2016-03-28.

[11] C. De Canniere, O. Dunkelman, and M. Knežević. KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 272–288. Springer, 2009.

[12] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[13] C. Miller and C. Valasek. A survey of remote automotive attack surfaces. *Black Hat USA*, 2014.

[14] N. F. Pub. 197: Advanced encryption standard (AES). *Federal Information Processing Standards Publication*, 197:441–0311, 2001.

[15] C. Solomon and B. Groza. Limon - lightweight authentication for tire pressure monitoring sensors. In *1st Workshop on the Security of Cyber-Physical Systems (affiliated to ESORICS 2015)*, 2015.