Designing Wireless Automotive Keys with Rights Sharing Capabilities on the MSP430 Microcontroller

Bogdan Groza, Tudor Andreica and Pal-Stefan Murvay
Faculty of Automatics and Computers, Politehnica University of Timisoara, Romania

Keywords: Automotives, RF Keys, MSP430.

Abstract: We explore the ultra-low-power microcontroller MSP430 from Texas Instruments as potential platform for developing vehicle keys. Radio frequency (RF) keys are still a relevant research subject as they are a common target for adversaries while automotive manufacturers show an increased interest in adding new functionalities to traditional keys while keeping them inexpensive. MSP430 is a low-cost, ultra-low-power, 16-bit capable microcontroller which can handle some cryptographic primitives that can be further used for designing secure authentication protocols. In this work we do explore the design and implementation options for a protocol that can be deployed in a car-sharing scenario where multiple users can share or gain access rights to the same vehicle. Due to inherent constraints of our platform, we keep the protocol simple and rely on inexpensive symmetric key primitives while still providing advanced options, e.g., rights sharing capabilities.

1 INTRODUCTION AND RELATED WORK

Despite their apparent simplicity and a rather long development history, car keys are still an interesting research subject. This happens because of several reasons. First, breaking car keys remains an attractive subject for hackers due to the inherent value of cars and nonetheless of the personal belongings inside the car. As cars are frequently left unattended in remote locations they are easily accessible to adversaries. Second, while increasing security is always possible by using more expensive technologies (e.g., SoA cryptographic designs, etc.) car manufacturers are frequently trying to cut on production costs. Thus one needs to achieve security at the lowest possible price. Third, in the recent years there has been a spectacular growth in functionalities that are present inside cars, due to the massive growth of technologies that are helpful to the driver and also entertaining for passengers. Remote controlling such functionalities becomes an immediate necessity and the vehicle key can be the link to these functionalities.

In this work we try to answer these necessities by exploring the MSP430 platform from Texas Instruments as potential platform for the development of RF vehicle keys. The main motivation comes from the fact that MSP430 is a low-cost, ultra-low power device that is widely available on the market. Nonetheless, it is a capable 16-bit controller with enough memory to handle basic cryptographic primitives that are mandatory for the design of a security protocol.

1.1 Targeted Scenario

The targeted scenario is depicted in Figure 1. The scenario that we target is a car sharing scheme where a user obtains access to a vehicle from an authorized car-sharing center and has the ability to delegate part of his rights further. We consider that the car sharing center can advertise the position of existing cars, i.e., their GPS coordinates, via a smart-phone based application (car-sharing by smart-phone apps is a realistic scenario and such schemes are already deployed in practice as several third-party solutions can be found on the web).

However, rather than relying on access via smartphones, which are not yet a perfect platform in terms of security, we want to rely on microcontroller-based RF keys which cannot be compromised by malicious applications. That is, a user that already has a car-key from the car-sharing center, can locate all available cars, choose the one that is in his closest vicinity and use it if he has the rights. Requests for the particular vehicle can be made from the smart-phone application, but further access is gained with the RF key.

Moreover, we want a key to express the ability to delegate a subset of existing rights to another key wit-

Groza, B., Andreica, T. and Murvay, P-S.
Designing Wireless Automotive Keys with Rights Sharing Capabilities on the MSP430 Microcontroller.
DOI: 10.5220/0006280601730180
In *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2017)*, pages 173-180
ISBN: 978-989-758-242-4

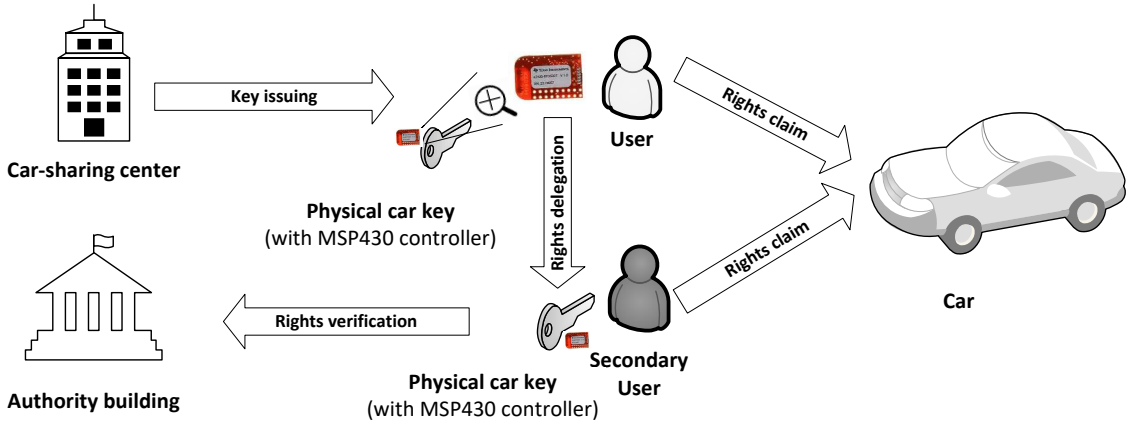

Figure 1: Car access-rights sharing scenario.

hout involving the sharing center. Such a functionality may be useful in case when a group rents a number of cars and then at some point decides to change cars between users or further keep a single car and reconfigure all keys to work with the particular car. To make access rights verifiable by neutral third parties, e.g., authorities, that must be able to trace back the key to the original user/issuer (for example in case of misuse) we do store digitally signed access rights on the key as soon as the user successfully claims his rights over the first car.

1.2 Related Work

Automotive RF keys were commonly analyzed by the research community and there are numerous results that showed them as insecure, e.g., (Francillon et al., 2011; Tillich and Wójcik, 2012; Verdult et al., 2012; Shoukry et al., 2013; Wetzels, 2014). Besides poor security, these classical car keys cannot answer to the demands of our practical scenario. They merely implement simplistic challenge-response protocols that are useless for right delegation and verification by neutral parties. Recently, the use of smart-phone-based technologies has been discussed (Busold et al., 2013), (Timpner et al., 2013), (Hong et al., 2016) and these approaches open road for complex car sharing procedures. However, our work tries to avoid the use of a secondary smart-device in order to save costs, remove the need for carrying a mobile phone and possibly increase security (as smartphones are easier to compromise). Consequently, our protocol design relies on distinct cryptographic constructions as we need to adapt to a platform with low computational capabilities.

2 PROTOCOL DESIGN

We begin by presenting the protocol design then proceed to a discussion on the choice of the cryptographic primitives.

2.1 Protocol Details

The proposed protocol suite is summarized in Figure 3. Briefly, the role of the protocol components is the following:

1. *Key issuing* is the procedure designated for releasing a new car key. This happens at the car sharing center ShCenter and we assume that it happens in a secure environment. Therefore, no security measures are needed on the channel. By this procedure, each key receives an identifier $ID_{\text{RFKey}_{ID}}$, a secret key shared with the sharing center $K_{\text{RFKey,ShCenter}}$, the access rights $\text{rights} = \{access, time, ltime, \text{owner:ShCenter}\}$ and the signature from the sharing center over the key identifier and rights $\text{sig\_rights} = Sig_{\text{ShCenter}}(ID_{\text{RFKey}}, \text{rights})$. The sharing center also stores freshness related information such as the time of registration and the lifetime of the user access rights, i.e., *time* and *ltime*. While the key cannot perform time-synchronization with a remote server and cannot keep a secure timer (the key is dependent on battery which may be disabled by an adversary), we also keep this freshness related information on the key to be verified by third parties. Since access to the vehicle is granted with the consent of the sharing center, it is sufficient for the sharing center to keep a secure timer and the car can synchronize with it.

2. *Rights claim key-to-car* is the procedure in which the user is claiming his rights over a car. First, the car receives the identifier of the key ID_{RFKey}, the access rights rights along with their signature sig_rights. If the signature is not authentic or the lifetime of the key has expired, the protocol stops. A random value $\text{rnd}_{\text{RFKey}}$ is further used in the challenge-response protocol to ensure freshness. The car needs to contact the sharing center for establishing a shared cryptographic key and further sends all the information that was received from the key along with its own fresh random value rnd_{Car}. We assume that the communication channel between the car and the sharing center is secure. This is a natural hypothesis since modern cars are equipped with 3G/4G communication and SSL/TLS capabilities are already present on automotive grade embedded devices. Designing a security protocol for the communication channel between the car and the sharing center would be out of scope for our work. The sharing center verifies the access rights, note that it may be that the key acquired the rights from the sharing center or from another key as discussed in the last procedure procedure. In the first case the sharing center has to verify its own signature by using the public key, in the second it has to verify a MAC (Message Authentication Code) with the secret key that is shared with the initial owner of the access rights. The car receives the shared key with the RF physical key which is computed by the sharing center as $\mathcal{KD}\left(\text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}}, K_{\text{RFKey,ShCenter}}\right)$. Additionally, the car receives a signature over the granted access rights and the new owner-car pair, i.e., RFKey, Car. Whenever the access rights expire, the car will deny access to the user. Now the car answers in a challenge response manner with a MAC computed over the participant identities, the random values and the shared key $K_{\text{RFKey,Car}}$. The physical key can already compute this shared key as it already is in possession of $K_{\text{RFKey,ShCenter}}$, then verify the response and answer to the challenge with a MAC over the values in reverse order, i.e., $\mathcal{MAC}_{K_{\text{RFKey,Car}}}(ID_{\text{RFKey}}, ID_{\text{Car}}, \text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}})$.

3. *Car access* is the basic procedure in which a user asks for a particular functionality to the car, e.g., open doors, windows, start the engine, etc. A challenge-response protocol with the shared key is used to prove the identity of each participant. The key simply asks the functionalities that it wants to perform *func* and the car responds with some random material rnd_{Car}. Then the key replies to the challenge with a MAC computed via the secret shared key over the identities, functionalities, its own random material and the received random value, i.e., $\mathcal{MAC}_{K_{\text{RFKey,Car}}}(ID_{\text{RFKey}}, ID_{\text{Car}}, func, \text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}})$.

Table 1: Summary of notations.

RFKey	an radio-frequency vehicle access key
Car	the vehicle to which access is requested
ShCenter	the vehicle sharing center
ID	id associated to an RF key or car
raccess	access rights to the vehicle
func	functionalities (requested from vehicle)
rnd	random value
$\mathcal{Sig}$	digital signature
$\mathcal{KD}$	key-derivation process
$\mathcal{H}$	hash function
$\mathcal{MAC}$	message authentication code

4. *Rights delegation key-to-key* is the procedure in which the owner of one key, can designate a subset of his rights to another key. First, the key sends a request message with its own ID. The users have to verbally agree and check on the display of their keys that the key IDs are correctly set. The second key responds with the granted access rights *raccess*$'$ and a MAC over them denoted as $\text{sig\_rights} = \mathcal{MAC}_{K_{\text{RFKey,ShCenter}}}\left(\text{RFKey}_2, \text{rights}'\right)$. Finally, the requester confirms that the requested access rights have been received. For practical reasons, considering a rather classical interface for the key, the requester can simply push the buttons of the key corresponding to the access rights and these are to be confirmed by the other party. While the physical design of the key is out of scope for our work, for clarity, we do suggest it in Figure 2. We consider that the insecure RF channel of this step is protected by the visual channel between the user and the key. Since there is no secretly shared value between the two keys, relying on user's feedback is the only alternative.

5. *Verification* by neutral third parties, e.g., authorities, is not presented as a distinct protocol component as this procedure is straight-forward from the certification chain. The neutral third party needs to be in possession of the public-key certificate of the sharing center and the RF key must externalize the signed access rights by the sharing center. This can be straight-forwardly achieved.

Figure 2: Suggestive depiction of the envisioned RF key from our work.

2.2 Choice of Cryptographic Primitives

Given the limited amount of memory, i.e., 1KB of RAM and 32KB of Flash (part of which is also needed for other non-security related tasks) and the absence of cryptographic hardware, symmetric key cryptography seems to be the only alternative. Symmetric primitives can be efficiently performed on MSP430 as will be detailed in the next section. The protocol relies only on simple hash functions and immediate derivatives: MACs and key-derivation functions.

More flexibility can be achieved by the use of public-key primitives, but these are for the moment too expensive for our setup. We give here only a brief discussion on this alternative. Asymmetric primitives, such as the RSA or DSA, do not appear to be an option due to technical limitations. This scenario may be interesting for reviving a cryptographic construction that today is somewhat out-of-focus: one-time signature schemes. While several limitations exists we briefly discuss here the possibility to use them. Such schemes were proposed in the '80s specifically for highly constrained devices. While there are also several more recent proposals, none of them shows dramatic improvements in performance compared to the simplest construction which is the genuine Merkle digital signature scheme (Merkle, 1988). The main drawback of Merkle-like signatures is that they require significant storage for each of the signed bits.

One improvement of the Merkle scheme (see Note 11.95 from (Menezes et al., 1996)) offers a time-memory trade-off that requires roughly $2\lambda\lceil k/\log_2\lambda\rceil$ computations and a signature of size $2\lceil k/\log_2\lambda\rceil$ hash outputs for signing k bits. Considering that the signature is performed over a hash (i.e., the traditional hash-then-sign paradigm) and that for short-term use 80 bits of security are sufficient, we try to get some rough computational estimates. For example, by setting $\lambda = 16$ the 80 bit output to be signed leads to a signature of $2*80/\log_2 16*80 = 2*20*80 = 3200$ bits. The number of computations is $2*16*80/\log_2 16 = 2*16*20 = 640$ hash function computations. By using in advance some of the computational results from the following section, the computational time is under 1 second. Consequently, using one-time signatures may be a possible alternative.

The only limitation is the one-time nature of such signatures. Since the aforementioned signature will require a key of 3200 bits and the controller has a memory of 32KB, assuming that at most half of the memory is used by the application data (a realistic assumption), 4 such keys can be stored in memory. By the use of Merkle trees these can be efficiently linked to an original key, thus signing multiple times is possible. While theoretically Merkle trees can be extended indefinitely, this will not be possible since storing the entire path of a signature will require too much memory. Thus, in case of one-time signatures, the sharing capabilities will be restricted to several sharing operations with the controllers of our setup. While we do rely on the simpler MAC-based protocol, one-time signatures may be considered for further extensions.

3 TECHNICAL DETAILS AND EXPERIMENTAL RESULTS

We first clarify the computational capabilities behind MSP430 microcontrollers, then we proceed to some details on the proposed network setup and provide results on energy consumption which are relevant for a RF key that relies on a small battery.

3.1 Computational Capabilities Behind MSP430

The 16-bit MSP430 family of microcontrollers features a multitude of configurations for a wide range of applications which require low power consumption including automotive products and wireless sensor networks. The computational efficiency of various cryptographic primitives running on this platform was presented in several works. Some computational results on running SHA1 and SHA2 on an MSP430 are given in (Romann and Salomon, 2014) while (Buhrow et al., 2014) presents speed and energy efficiency measurements for several versions of AES and Speck. A number of papers (Hinterwälder et al.,

Key issuing (secure environment at car-sharing center)

1. ShCenter ⇢ RFKey: $ID_{\text{RFKey}}, K_{\text{RFKey,ShCenter}} = \mathcal{KD}(K_{\text{master}}, ID_{\text{RFKey}})$,
   $\text{rights} = \{raccess, time, ltime, \text{owner:ShCenter}\}, \text{sig\_rights} = Sig_{\text{ShCenter}}(ID_{\text{RFKey}}, \text{rights})$

Rights claim key-to-car (insecure environment)

1. RFKey → Car: $ID_{\text{RFKey}}, ID_{\text{Car}}, \text{rnd}_{\text{RFKey}}, \text{rights}, \text{sig\_rights}$
2. Car ⇢ ShCenter: $ID_{\text{Car}}, ID_{\text{RFKey}}, \text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}}, \text{rights}, \text{sig\_rights}$
3. ShCenter ⇢ Car: $K_{\text{RFKey,Car}} = \mathcal{KD}(\text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}}, K_{\text{RFKey,ShCenter}})$,
   $Sig_{\text{ShCenter}}(ID_{\text{RFKey}}, ID_{\text{Car}}, \text{rights})$
4. Car → RFKey: $ID_{\text{Car}}, ID_{\text{RFKey}}, \text{rnd}_{\text{Car}}, \mathcal{MAC}_{K_{\text{RFKey,Car}}}(ID_{\text{Car}}, ID_{\text{RFKey}}, \text{rnd}_{\text{Car}}, \text{rnd}_{\text{RFKey}})$,
   $\text{sig\_car\_rights} = Sig_{\text{ShCenter}}(ID_{\text{RFKey}}, ID_{\text{Car}}, \text{rights})$
5. RFKey → Car: $ID_{\text{RFKey}}, ID_{\text{Car}}, \mathcal{MAC}_{K_{\text{RFKey,Car}}}(ID_{\text{RFKey}}, ID_{\text{Car}}, \text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}})$

Car access (insecure environment)

1. RFKey → Car: $ID_{\text{RFKey}}, ID_{\text{Car}}, func, \text{rnd}_{\text{RFKey}}$
2. Car → ShCenter: $ID_{\text{Car}}, ID_{\text{RFKey}}, \text{rnd}_{\text{Car}}$
3. RFKey → Car: $ID_{\text{RFKey}}, ID_{\text{Car}}, \mathcal{MAC}_{K_{\text{RFKey,Car}}}(ID_{\text{RFKey}}, ID_{\text{Car}}, func, \text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{Car}})$

Rights delegation key-to-key (insecure environment [a])

1. RFKey_2 → RFKey_1: ID_{RFKey_2}, *rightsRequest*
2. RFKey_1 → RFKey_2: $ID_{\text{RFKey}_1}, ID_{\text{RFKey}_2}, \text{rights}' = \{raccess', time', ltime', \text{owner:RFKey}_1\}$,
   $\text{sig\_rights} = \mathcal{MAC}_{K_{\text{RFKey,ShCenter}}}(\text{RFKey}_2, \text{rights}')$
3. RFKey_2 → RFKey_1: $ID_{\text{RFKey}_2}, ID_{\text{RFKey}_1}, \text{rights}'$

[a] users will have to check and confirm on the visual display of the physical key that rights are shared between the correct identities RFKey_1 and RFKey_2

Figure 3: Components of the proposed protocol suite: key issuing, rights claim, car access and rights delegation.

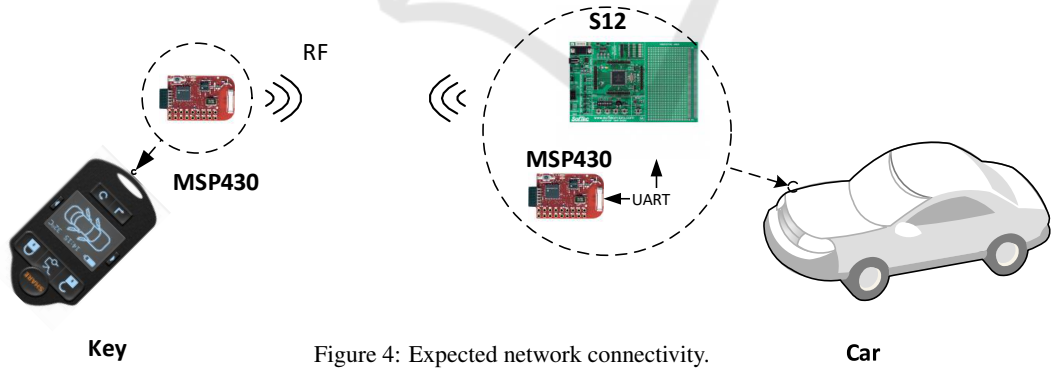

Figure 4: Expected network connectivity.

2014), (Szczechowiak et al., 2008), (Wenger and Werner, 2011) focus on the implementation and evaluation of elliptic curve cryptography on MSP430 family members proving the feasibility and limits of such approaches on this group of constraint devices.

We selected the MSP430F2274, a member of the MSP430F2X/4X subfamily, as a candidate platform for implementing the proposed wireless access system. Versions of the MSP430F227 are available with aerospace qualifications which are even more strict than the ones in the automotive domain. With its 1KB of RAM, 32KB of Flash and a maximum operating frequency of 16MHz our choice fits the device category used for automotive key applications.

In previous work (Murvay et al., 2016) we evaluated the capabilities of this device for executing cryp-

Table 2: Execution speed (milliseconds) on MSP430 at 16MHz of some hash functions based on previous work (Murvay et al., 2016).

Input size (bytes)	Cryptographic primitive (block size)			
	MD5 128	**SHA1 160**	**SHA2 256**	**Blake2 256**
8	0.427	3.642	3.691	2.263
64	0.709	7.304	7.117	2.270
576	2.985	36.74	34.50	16.85
1536	7.253	91.95	85.85	44.19
4096	18.63	239.2	222.8	117.1

tographic primitives. The test results showed that the MSP430 core can successfully handle symmetric cryptographic primitives as proved by measurements illustrated in Table 2 which presents the execution speed for several hash functions [1] when executed for various input sizes at an operating frequency of 16MHz. Note that here we refer to the performance of the core since the results are only dependent on the operating frequency and are indicative on the computational capabilities of the entire MSP430 family when the same frequency is employed.

Code size for each hash function implementation is provided in Table 3 along with the Flash memory occupation percentage relative to the MSP430F2274 memory size. The evaluated primitives occupy less than 20% of the Flash memory area when no optimizations are applied leaving sufficient space for implementing the communication protocol and other application features. If needed, occupied space could be decreased by applying code size optimizations.

Table 3: Flash memory consumption of primitive implementations on MSP430 based on previous work (Murvay et al., 2016).

MD5 128		SHA1 160		SHA2 256		Blake2 256	
bytes	%	bytes	%	bytes	%	bytes	%
6394	19.98	1338	4.18	2610	8.16	4046	12.64

Since the available RAM memory is not sufficient both for program execution and for storing cryptographic keys the Flash memory remains the only storage alternative in the absence of an EEPROM. The amount of Flash memory available for key storage is limited by the space required for storing the actual program object code and by the foreseen number of required re-writes of the key storage space during the lifetime of the product. The MSP430 Flash is guaranteed for 10.000 erase cycles. If more re-write cycles are required this can be achieved by EEPROM emulation at the cost of a significantly reduced storage space depending on the key storage structure.

Our experimental setup consists of two MSP430 Wireless Development Tools(EZ430-RF2500) made by Texas Instruments and a NXP ZK-S12-B Kit. The two MSP430 boards are used both for wireless communication as well as for the proposed protocol implementation. One of them is integrated in the car key, while the other one is connected to the car's BCM. The S12 board contains a MC9S12C128 microcontroller and is used to stand for the BCM.

3.2 Details on Connectivity

It is relevant to point out that designing the key alone is not sufficient without the corresponding counterpart inside the vehicle. Thus, the network design that we consider consists in a secondary MSP430 controller that is placed inside the car and communicates with the BCM (Body Control Module) of the car, a unit which is responsible to all functionalities related to the body (windows, doors, trunk, etc.). For illustration purposes we choose a board equipped with a Freescale S12 core, a controller that is used in real-world BCMs. We find that the communication between the two devices can be easily done by using the UART (Universal Asynchronous Receiver/Transmitter) interface. For our experimental setup we chose to use a baud-rate of 4.8 kBaud, but the baud-rate can be configured anywhere in a range between 1.2 and 38.4 kBaud. This setup is suggested in Figure 4.

A second detail on connectivity is that traditional vehicle RF keys operate in the Ultra high frequency (UHF) at 433 MHz. Microcontrollers from the MSP family do posses sub-1GHz communication capabilities. Our MSP430 experimental devices do communicate in the 2400-2483.5 MHz band. Using this frequency band is not an issue in general as it is already used in the automotive domain, e.g., for car alarms. Moreover, all of the present results will hold on any other MSP-based system core that uses a sub-1GHz transceiver.

3.3 Energy Consumption

Figures 5 and 6 depict the energy consumption, acquired by the use of an Agilent oscilloscope, as the MSP430 microcontroller transits between several states including a hash function computation state for SHA1 in the former figure and SHA256 in the latter.

The voltage scale is at 50mV per division while the time scale was set to 1s per division. The inten-

[1] MD5 and SHA1 are known to be insecure and we keep them just as a bottom line for performance

tion is to highlight the power consumption for cryptographic functions in contrast to other functionalities. In the first state (1), the microcontroller is in low-power mode, then in the second state (2) the controller is in normal mode and runs no tasks in the background. This is followed by a third state in which an LED is lit (3) and the fourth state in which it executes the corresponding hash function (4). Then in the last state (5) several messages are sent via RF. It is easy to see that the energy consumption for executing the cryptographic primitive is very low compared to the RF transmission or even lighting a 2mA rated LED. Consequently, the cryptographic functionalities will not have a more significant contribution to the battery depletion rate than regular tasks, e.g., reading a button or making an LED blink. There is no noticeable difference between the power consumption recorded during the computations of the two hash functions which . Moreover, executing SHA1 and SHA256 on the MSP430 platform is very similar in terms of duration (e.g. a hash execution on an 8 byte input takes 455μs for SHA1 and 461μs for SHA256). This makes the two plots look very similar although they represent execution cycles involving two different hash functions.

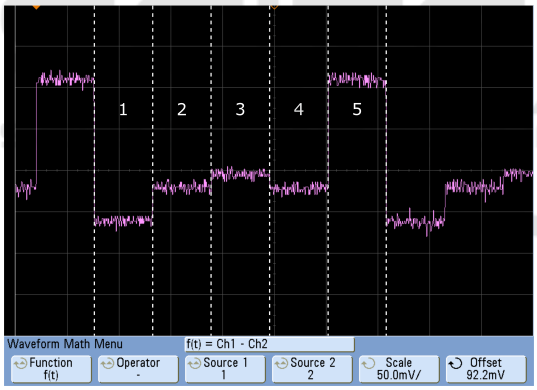

Figure 5: Energy consumption during SHA1 computations (4) compared with the consumption obtained for periods of low-power mode, normal mode without tasks, turned-on LED and during the RF transmissions.

4 CONCLUSION

Our work sets the first steps in the use of the MSP430 platform from Texas Instruments for vehicle RF keys. So far the results prove that basic security functionalities (e.g., symmetric functions, challenge-response protocols) are straight-forward to deploy while more advanced security functionalities (e.g., digital signatures) are also within reach. The main scope of our

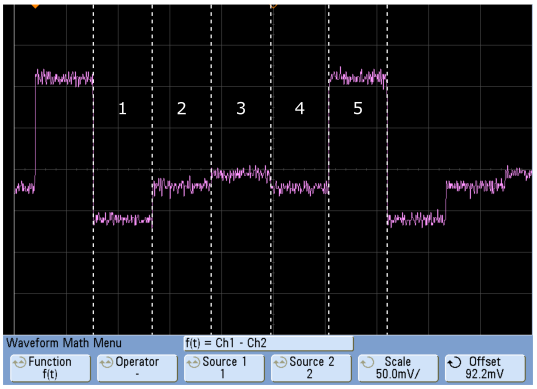

Figure 6: Energy consumption during SHA256 computations (4) compared with the consumption obtained for periods of low-power mode, normal mode without tasks, turned-on LED and during the RF transmissions.

work was to clarify some of the technical constraints on MSP430 raised by the proposed scenario. A full-scale implementation, security proofs for the presented schemes, as well as their redesign in case of flaws, is a relevant subject for future work in case that the ideas prove promising for further investigations.

ACKNOWLEDGEMENTS

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project number PN-II-RU-TE-2014-4-1501 (2015-2017).

REFERENCES

Buhrow, B., Riemer, P., Shea, M., Gilbert, B., and Daniel, E. (2014). Block cipher speed and energy efficiency records on the MSP430: System design trade-offs for 16-bit embedded applications. In *International Conference on Cryptology and Information Security in Latin America*, pages 104–123. Springer.

Busold, C., Taha, A., Wachsmann, C., Dmitrienko, A., Seudié, H., Sobhani, M., and Sadeghi, A.-R. (2013). Smart keys for cyber-cars: Secure smartphone-based NFC-enabled car immobilizer. In *3rd ACM Conference on Data and Application Security and Privacy*, pages 233–242. ACM.

Francillon, A., Danev, B., Capkun, S., Capkun, S., and Capkun, S. (2011). Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*.

Hinterwälder, G., Moradi, A., Hutter, M., Schwabe, P., and Paar, C. (2014). Full-size high-security ECC implementation on MSP430 microcontrollers. In *International Conference on Cryptology and Information Security in Latin America*, pages 31–47. Springer.

Hong, J., Shin, J., and Lee, D. (2016). Strategic management of next-generation connected life: Focusing on smart key and car–home connectivity. *Technological Forecasting and Social Change*, 103:11–20.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Merkle, R. C. (1988). A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, pages 369–378, London, UK. Springer-Verlag.

Murvay, P.-S., Matei, A., Solomon, C., and Groza, B. (2016). Development of an AUTOSAR Compliant Cryptographic Library on State-of-the-Art Automotive Grade Controllers. In *Proceedings of the 11th International Conference on Availability, Reliability and Security, ARES*.

Romann, R. and Salomon, R. (2014). Salted hashes for message authentication-proof of concept on tiny embedded systems. In *Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*, pages 42–46. IEEE.

Shoukry, Y., Martin, P., Tabuada, P., and Srivastava, M. (2013). Non-invasive spoofing attacks for anti-lock braking systems. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 55–72. Springer.

Szczechowiak, P., Oliveira, L. B., Scott, M., Collier, M., and Dahab, R. (2008). *NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks*, pages 305–320. Springer Berlin Heidelberg.

Tillich, S. and Wójcik, M. (2012). Security analysis of an open car immobilizer protocol stack. In *Trusted Systems*, pages 83–94. Springer.

Timpner, J., Schürmann, D., and Wolf, L. (2013). Secure smartphone-based registration and key deployment for vehicle-to-cloud communications. In *Proceedings of the 2013 ACM Workshop on Security, Privacy and Dependability for CyberVehicles*, pages 31–36. ACM.

Verdult, R., Garcia, F. D., and Balasch, J. (2012). Gone in 360 seconds: Hijacking with hitag2. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 37–37. USENIX Association.

Wenger, E. and Werner, M. (2011). Evaluating 16-bit processors for elliptic curve cryptography. In *International Conference on Smart Card Research and Advanced Applications*, pages 166–181. Springer.

Wetzels, J. (2014). Broken keys to the kingdom: Security and privacy aspects of rfid-based car keys. *arXiv preprint arXiv:1405.7424*.