LiMon - Lightweight Authentication for Tire Pressure Monitoring Sensors

Cristina Solomon and Bogdan Groza

Faculty of Automatics and Computers,
Politehnica University of Timisoara, Romania
cristina_solomon@ymail.com
bogdan.groza@aut.upt.ro

Abstract. Modern vehicles offer a raw territory for designing security solutions as the over-increasing design complexity demanded massive advances in electronics in the absence of a crisp vision over the adversary model. The vehicle Tire Pressure Monitoring System (TPMS) is a sub-system that recently triggered some attention in the light of several reported attacks. In this work we start from analyzing existing proposals and reckon some shortcomings, e.g., academic proposals are not yet tested on real-world components while a patented security solution from the industry (likely deployed in practice) is completely insecure. Motivated by these, we design a new solution and deploy it on real-world components that are used in the automotive industry. Designing security for this subsystem proves to be especially relevant as the computational resources for TPM systems are somewhat at the minimum to be found in automotive embedded devices. Our solution is deployed on Infineon SP37 sensors and takes advantage of some recently proposed light-weight cryptographic designs, e.g., SPECK and PRESENT.

Keywords: authentication, wireless sensors, vehicles

1 Intoduction

Contemporary vehicles are a powerful example of a cyber-physical system with advanced functionalities, complex electronics, numerous communication interfaces and yet only sparse security layers. The urgent demand for adding security is proved by many of the recently reported attacks [14], [5] to which so far there was little response from the industry.

The standard functionality for tire sensors is to monitor the pressure inside the tire which further impacts on at least three directions. First, it reduces tire wear and fuel consumption, resulting in cost efficiency on the driver's side. Second, it reduces CO2 emissions comforting the environment. Third, it improves the breaking distance and the control of the vehicle, thus having a positive effect on the safety of drivers and passengers. However, it is worth mentioning that a recent study showed the benefits of TPMS in reducing CO2 emissions to be only marginal [17]. Still, for safety and comfort the advantages of the system are undisputed. Besides pressure monitoring, the system usually monitors the temperature and acceleration. Future functionalities may include monitoring for tire wear and adapting driving characteristics to the tire specifications

(e.g., winter vs. summer tire). The TPMS system became mandatory in the US since 2008 and in the EU since 2014.

Generally, there are two distinct kind of implementations: i) direct systems (the common choice and the subject of this work) in which a sensor is located inside the tire and ii) indirect systems which rely on information from the ABS (improperly inflated tires lead to detectable differences when breaking, but such systems are inaccurate and not commonly used). While some TPMS deployments were initially based on four receivers, one nearby each wheel, the current trend is to have a single receiver that gathers the frames from each wheel sensor. This lowers the production cost, but has the additional disadvantage that the system has to learn or be set to recognize the exact wheel from which the data originates. This association is done either by active learning, e.g., by correlating the acceleration reported by the sensor with the acceleration independently reported by the ABS system on each wheel, or it is set in an off-line manner with a diagnosis tool based on the ID of each sensor.

SECURITY ISSUES. The TPMS is considered part of the safety system of the car since improperly inflated tires can lead to accidents with catastrophic consequences. Still, security mechanisms are completely absent from such systems and recent research works [12] were quick in determining several attacks on such platforms: eavesdropping can be easily performed at a range of 10m (and with improved instrumentation at up to 40 m), vehicles can be tracked by using the ID that is broadcast by the sensors, packets can be injected in the system, triggering false alarms on the display (beside the obvious discomfort, these may determine the driver to stop the car for inspection which could cause further incidents), battery drainage is possible if there are packets that could trigger an immediate response from the sensors. The attacks presented in [12] require just an average adversary that is in possession of some easy to find radio equipments. Further capabilities of the adversary such as side-channel attacks can be mentioned, but these do not appear to be a serious concern yet.

While so far the information from the TPMS system has only a passive (informative) role, the recent patent application for TPMS security from Continental [16] proves that the industry is aware and concerned by such attacks. The patent suggests that the main concern is on the fact that the driver can be determined to stop the car in case that an adversary injects data claiming that the tires are under-inflated [16] (a similar scenario is also outlined in the work that initially showed these attacks [12]). Clearly, threats can become much more serious in the near future if information from the TPMS sensors will be also actively used by breaking or stability controls inside the car. As a consequence to these, opening road for research in this direction is clearly necessary.

2 Analysis of previous proposals

To the best of our knowledge there are only two previous proposals for assuring security in TPMS sensor systems. The first is a proposal from the academic research community [18] and the second is the patent application from Continental [16] which may be already deployed on the market.

We discuss both these proposals in what follows but let us note from the beginning that, probably not surprising, the proposals are situated on two extremes. The patent [16]

tries to build an inexpensive solution by garnering what already exists on the sensors but fails to result in a secure protocol. Xu et al. [18] provide a rigorous security design but the experimental results are obtained on an Arduino platform which makes it unclear if the solution can be readily deployed on real world TPM systems.

CONTINENTAL'S PATENT EP2719551 [16]. On the positive side, this appears to be the first attempt from the industry to address these security issues. At the very least the patent shows that the industry is aware of the problem and the solution can be viewed as an improvement to the no-security option. On the negative side, the solution completely fails in assuring security as the proposed mechanism can be trivially broken, moreover, the brief security analysis provided in the patent application is wrong (as we discuss next). The solution works as follows: two PRNGs are used (both implemented around a CRC polynomial) one for the wheel unit, the other for the receiver. Since there are four wheels, each wheel has its own PRNG built on the same CRC code but initialized with a distinct seed, while the receiver keeps 4 instances, one for each of them. A seed, generated from data that is exchanged between the wheel units and the receiver at vehicle start-up, is used as input to the PRNG and the output (referred as authentication marker) is XORed with the CRC of each message in order to assure authentication. We now enumerate the problems in this proposal starting with the more severe ones:

1. CRCs (Cyclic Redundancy Checks) are used to build an *authentication marker*. It is commonly known that while CRCs can assure integrity checks (against unintentional errors) they fail in assuring authenticity (even if one embeds a secret key in the construction) because they are linear transformations. Each packet sent from the wheel unit has its CRC XORed with an authentication marker that is generated by another CRC over a secret seed, i.e., $data||\mathrm{CRC}(data) \oplus \mathrm{CRC}(seed)$. Obviously, this protection is useless, since CRCs are linear transformations one can simply XOR the genuine data with a distinct value and do the same for the CRC and the packet will be valid, i.e., $data \oplus data'||\mathrm{CRC}(data) \oplus \mathrm{CRC}(seed) \oplus \mathrm{CRC}(data')$. Thus breaking the scheme is effortless and does not require to find the secret key at all.
2. The secret seed can be easily found. The patent is not really clear on the size of the underlying CRC code, however under paragraph [0034] it claims that it can be very light, 8 bits being sufficient since for an emission period of 60 seconds since to observe the full period of 256 rounds one would need 4.25 hours. This claim actually misses the fact that the CRC, being a LFSR, can be broken just after witnessing a sequence equal to its size, in this case 8 packets. This makes the time required to gather enough information for recovering the seed at around 8 minutes instead of 4.25 hours.
3. The key-sharing procedure for exchanging the seed between the wheel units and the central receiver has no protection at all. Since there is no cryptography implemented on the nodes, the seed value is exchanged unencrypted when the system starts. This makes it easy to mount an attack if the adversary witnesses the communication right from the beginning. But in the light of the previous attacks, this would be a minor issue.

4. A final problem are the ambiguities in the patent. For example the patent claims that "at least a part of each data packet is encoded with an authentication marker" but fails to make it clear which part. Since such information can be easily revealed by a simple analysis of the packet we believe that such ambiguities were not demanded by technical concerns (i.e., making the mechanism more secure, since in fact it isn't at all secure) but rather by legal issues (i.e., making the patent cover a larger area of applications). The same ambiguities are in describing the exact security level that is expected, i.e., the size of the secret seed, the CRC polynomial, etc. While avoiding such details may help in extending the legal range of the patent, this solution is insecure and it is highly unlikely for it to have any practical benefits.

XU ET AL. [18]. The protocol is well designed, particularly targeting spoofing and tracking attacks. The core of the protocol is built around the 32-bit symmetric encryption algorithm KATAN and uses CBC-MAC to assure message authentication. Additionally, a LFSR is used to generated sequential numbers that look pseudo-random in order to prevent replay-attacks (the use of a simple counter is avoided in order to prevent tracking of the device, as sequence numbers are predictable). To ensure privacy, as vehicles can be tracked by the use of the sensor's ID, a pseudo-ID is randomly generated before the generation of each session key. A proof-of-concept implementation is provided on an Arduino based platform. The shortcoming that we see for this proposal is that it is deployed on a platform that does not matches the real-world deployments, our experimental analysis showed that KATAN is too intensive for real-world TPMS sensors. We give more details on the real-world system specification in the following section.

3 Experimental setup

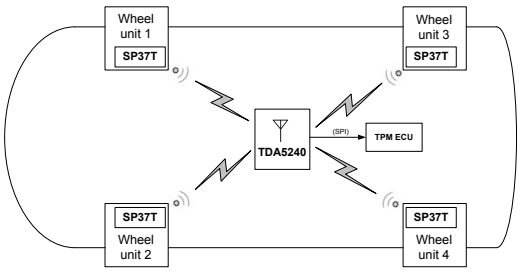

Fig. 1. TPM system with Infineon SP37 sensors and TDA5240 receivers

The devices used in our experiments are two dedicated TPMS development kits from Infineon. The first is the SP37T development kit [9], used for programming the TPMS sensors. This is a real-world TPMS sensor used by leading car manufacturers.

Characteristic	ATmega328P	Infineon's SP37 (8051 based)
Operating Voltage	5V	3.3V
Supply voltage range	1.8 - 5.5V	1.9 - 3.6V
Digital I/O Pins	14	19
Analog Input Pins	6	8
DC Current per I/O Pin	40 mA	10 mA
Flash Memory	32 KB	**6 KB**
RAM	2 KB (SRAM)	**256 Bytes**
EEPROM	1 KB	**31 byte emulated EEPROM** (+ 12 KB ROM)
Clock Speed	16 MHz	12 MHz
Temperature range	-40°C to 85°C	-40°C to 125°C

Table 1. Characteristics of ATmega328p vs. Infineon SP37 8051 sensors

It is based on a standard 8051 low performance microcontroller. Beside the microcontroller and the TPMS specific sensors, the development kit includes a LF receiver, a RF transmitter unit and an ADC converter for signal conditioning. Because power consumption is an essential aspect for a TPMS sensor, the SP37 sensor has a low power design and several low power operating modes. The power consumption peak lies in the RF transmission, which means that the shorter the sent telegrams are, the longer the battery lifetime will be. The SP37 sensor can be programmed to transmit the data frames in a cyclic fashion or based on requests.

The second development kit represents the receiver kit for TPMS messages, which is a SmartLewisRx TDA5240 [11] receiver kit. The TDA5240 chip is an enhanced Sensitivity Multi-Channel Quad - Configuration Receiver with Digital Baseband Processing, meaning that it can listen to up to 4 parallel sources, in our case, the TPM sensors. It has an autonomous receive mode, which reduces the noise of the host processor, improves the sensitivity and also reduces the power consumption of the system. The TDA5240 integrated circuit does not rely on a microcontroller, so it must be attached to an external one in order to evaluate and compute the received information from the sensors. This receiver is specifically designed for TPM systems.

The boards of the development kits (one with the SP37 sensor and the other one with the TDA5240 receiver chip) are connected through System Interface Boards (SIB) via USB to standard notebooks for development purposes. In case of the TDA5240, the controller from the SIB board represents the aforementioned external microcontroller. Figure 1 presents a TPMS deployment based on a central receiver that also corresponds to the setup addressed in our work. The technical specifications of the SP37 sensor are presented in Table 1 and compared with the Arduinno platform from [18].

The SP37 TPMS sensor is able to transmit the data frames at the 315 MHz or 433 MHz bands (UHF) and to employ ASK (Amplitude Shift Keying) or FSK (Frequency Shift Keying) modulation. The same configuration is supported also by the receiver TDA5240. In our test environment we use 433 MHz and FSK. Regarding the message encoding format, we use Manchester encoding.

The rough format of the data frames from the sensor is depicted in Figure 2. The frame contains 32 bits for the specific sensorID, and 16 bits for the temperature, acceleration, pressure measurements and battery status. The status data of the sensor is comprised by the unique sensorID and the internal battery status. The reported wheel

parameters are: acceleration, temperature and pressure (all measured within the sensor specific ranges [8]).

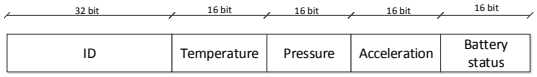

Fig. 2. TPMS data frame format with 16-bit data and 32-bit ID

For the TPMS sensor configuration and measurement, as well as for establishing the RF communication between the sensor and the receiver, we used the available SP37T ROM Library configuration and measurement functions provided with the SP37T development kit. These functions are described in the SP37T ROM library guide for which the reader is referred to [10].

4 Protocol design

Before getting to the protocol description, let us briefly caps on some of our design goals:

- *Lightweight cryptography.* We make use of some of the lightest symmetric primitives to assure real-time security. Ideally, we should rely on hardware implementations of current standards, e.g, AES. But since such an implementation is missing from the sensors, we make use of some of the lightest designs published so far SPECK and PRESENT. In the results section we provide quantitative measurements on the performance of software implementations for these primitives.
- *Optional confidentiality and privacy*. Keeping the information private from eavesdroppers at the cost of additional encryption operations does not appear to be the default option since privacy does not appear to be the main concern (it is energy consumption and authenticity that are more critical). We keep frame encryption, which implies hiding the ID as a second option in our protocol.
- *ISO based key-exchange.* While establishing a master key between each sensor and the receiver leads to a more surreptitious discussion, a fresh session key needs to be established regularly. Coming with a per-paper key-exchange protocol is not a desired option, especially when we address an area standardized by the industry. For this reason, we rely on well established ISO based key-exchange protocol.
- *Energy efficiency.* This is an obvious design goal, also expressed by the first two objectives, in addition to these it is clear that our protocol can rely only on symmetric cryptography. In the results section, we provide a crisp image on energy consumption.

4.1 Master and session key establishment

Before designing the authentication protocol between the wheel sensors and receiver, a shared key between the two needs to be established. This proves to be a difficult

issue because we cannot rely on public-key infrastructure due to the computational constraints at the sensor level. We first review the alternatives as they are proposed in related work.

Xu et al. [18] propose for the initialization of the secret master key to be triggered only by the sensor and only at events that are less likely to happen, e.g., when the tire is inflated for the first time. The procedure is also assumed to happen in some authorized garage, ensuring a safe environment for the key-exchange. The procedure is correct but it also bears a drawback. It seldom happens that drivers need to change the wheel on their own as the car is located hundreds of kilometres away from any authorized garage, or the driver may simply want to avoid the strains of finding an authorized garage. In this case the aforementioned procedure does not offer enough support. In Continental's patent [16], the key is exchanged in clear-text each time the vehicles starts. This procedure is easy to implement but also easier to eavesdrop, nullifying the security of the protocol that follows.

To bring a crisper image, we proceed further by classifying the key sharing alternatives in two categories:

1. *The resurrecting duckling.* The seminal work of Stajano [15] introduces the idea of imprinting by relating to a metaphor inspired from biology: a duckling emerging from its egg recognizes as its mother the first moving object it sees that makes a sound, a phenomenon called imprinting. The same idea is extended in [15] to devices: they are to be paired with the first entity that sends them the key. While the procedures in the Continental's patent [16] and the ones from the work of Xu et al. [18] may seem of distinct nature, they are in fact both particular cases of the *imprinting* phenomenon. Trying to refine all the procedures from [18] and [16], we can distinguish between three options.
   - *Rare event imprinting* is the case when the key is imprinted when a rare event takes place, e.g., wheels stopped rapidly at high speed (or oscillating clockwise and counter clockwise at a very low speed). Such events can happen only when the car (or wheel) is in special circumstances, e.g., on an elevator.
   - *External tool imprinting* is the case when the key is imprinted by the use of an external tool via a secure communication medium. This tool can be an OBD (On-Board Diagnosis) device that is paired with the driver's phone via a secure Bluetooth connection allowing easy configuration from the driver's side.
   - *Safe environment imprinting* requires the key to be imprinted in a secure environment, e.g., at an authorized garage. In this case the key can be simply sent in plaintext, for example when the wheel is used for the first time. Sending the key in plaintext each time the car starts, similar to the solution of the patent, cannot be considered a safe environment imprinting since there are no security guarantees for such a scenario.

   In all situations we assume that imprinting pairs the sensor with the first device that sends it a key. As noted in the work of Stajano [15] a reset procedure is also needed (this is in fact referred as *escrowed seppuku* meaning that the device can be killed in order to resurrect it for a new key). We believe that this procedure could also call either for a rare event, or for a master reset-key that is dedicated to this purpose. Likely, this master-reset key can be also imprinted by the manufacturer

and delivered to the owner in a closed envelope. This is indeed similar to the PIN vs. PUK code in mobile telephony. That is, the PIN is usually a short code chosen by the user, if the user loses the PIN then this can be reset by the longer PUK code that was imprinted by the manufacturer.

2. *Environmental data.* We should not forget a relevant aspect, all these sensors are located in the same environment which can provide a reach common entropy that is hard to be guessed by outsiders. First, accelerometer data is available both to the internal ABS unit as well to the wheel sensor. This data was previously used to learn the wheel on which a tire is connected, but can be as well used to generate a shared key. Generally, vibrations from the environment can be easily captured and they should be similar for devices located on the same vehicle. There is extensive literature on using accelerometer data to generate a shared key. Since this key will not provide an exact match on the sender and receiver side, fuzzy cryptography [6] can be applied to correctly extract a key.

To conclude on the master-key sharing procedure, we believe that choosing from one of the above depends mostly on the manufacturer and a more comprehensive analysis of the advantages and disadvantages is out of reach for our current work.

A fresh session key needs to be established each time the protocol starts or whenever the session counter reaches the maximum value, e.g., 0xFFFF in case of 16 bit counters. Given the industry driven nature of the application area, it is best to stay to current standards in order to make such solutions viable for adoption by manufacturers. The ISO/IEC 9798 provides a well known family of authentication protocols based on both symmetric and asymmetric primitives. Recently, this standard was subject to rigorous formal analysis which lead to several improvements and fixes [1]. Once the master-key is shared between the sensor and receiver, the session key-establishment can be done with any mutual authentication protocol. For example, the ISO 9798-2-4 three-pass mutual authentication scheme. In Figure 3 we depict the structure of this key-exchange protocol. The notations are the expected ones: $\mathbb{K}_{ms}$ denotes the master key shared between the sensor and the receiver, N is the regular notation for cryptographic nonces, i.e., random values, while $\mathsf{Sens_{ID}}$ and $\mathsf{Receiver}$ are the identities of the sending sensor and of the central receiver. The session key $\mathbb{K}_{ses}$ can be simply derived from the exchanged values $N_{\mathsf{Sens_{ID}}}$, N_{Receiver} and the master key $\mathbb{K}_{ms}$ with the help of some key derivation process. This incurs only a small computational cost in the order of an additional encryption and can be straight-forwardly built upon the primitives that we deploy for the rest of the protocol.

4.2 Frame authentication and encryption

Having fixed the the session key $\mathbb{K}_{ses}$ we again derive by some standard key derivation technique the authentication key $\mathbb{K}_{aut}$ and the encryption key $\mathbb{K}_{conf}$.

Frame authentication is done with the standard CBC-MAC based on SPECK as outlined in Figure 4 (i). This construction is secure as the data field has a fixed size of 64 bits in our example (the length of the full frame is 96 bits since 32 bits are added for the authentication tag). The 8-bit data fields suggested in Figure 4 serve us only as a baseline, giving the minimum expected size for such frames. We decided to scale down the

Handshake for session-key establishment

1. $\mathsf{Sens_{ID}} \rightarrow$ Receiver: $\mathsf{Sens_{ID}}, N_{\mathsf{Sens_{ID}}}$
2. Receiver $\rightarrow \mathsf{Sens_{ID}}$: $\{N_{\mathsf{Sens_{ID}}}, N_{\mathsf{Receiver}}, \mathsf{Receiver}\}_{\mathbb{K}_{ms}}$
3. $\mathsf{Sens_{ID}} \rightarrow$ Receiver: $\{N_{\mathsf{Receiver}}, N_{\mathsf{Sens_{ID}}}\}_{\mathbb{K}_{ms}}$

Fig. 3. Example of session key establishment based on ISO 9798-2-4 three-pass mutual authentication based on symmetric primitives

16 bits data fields to 8 bits since this resolution should be enough for most applications and will significantly improve energy consumption. If larger data is collected, then the fields can be extended to 16-bits without impacting the authentication tag which stays at 32 bits, however this will require to switch to a larger block size for SPECK as well.

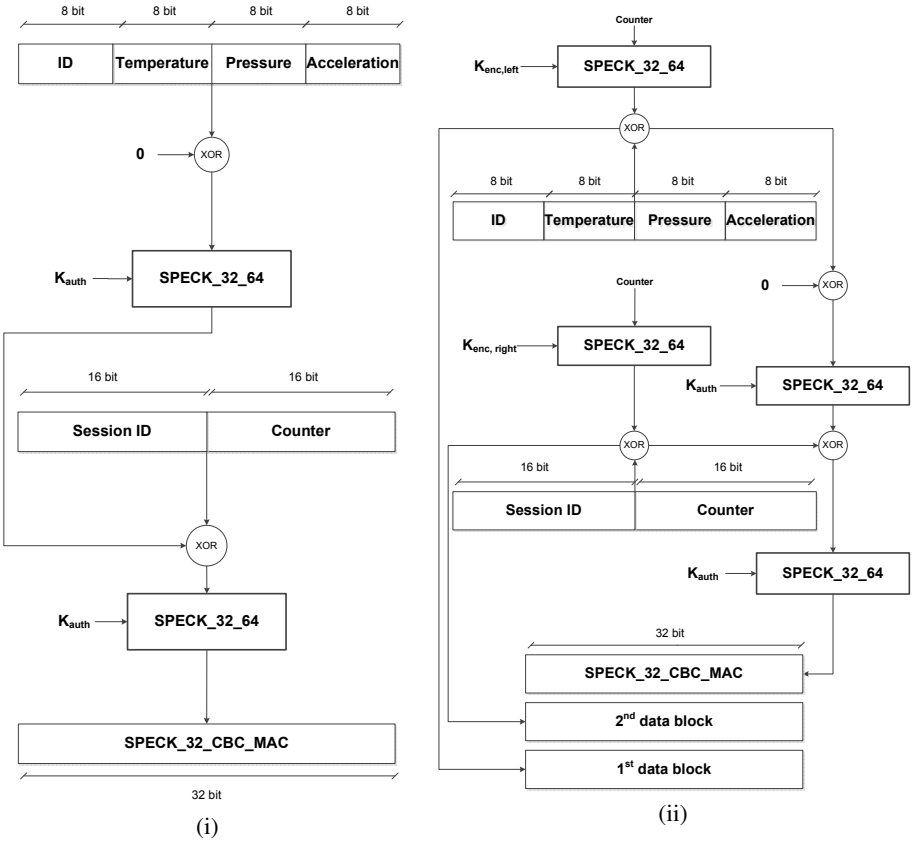

Fig. 4. A 96 bit frame: i) authenticated with CBC-MAC and ii) with authenticated encryption (encrypt-then-MAC) based on SPECK CBC-MAC)

Frame encryption is added only as an option but not as part of the default protocol description as outlined in Figure 4 (ii). The reason behind this is that the benefit of encrypting the frames seem to be somewhat marginal. The main intention is to assure privacy for the user, e.g., the ID of the sensors is hidden and thus it cannot be tracked (this is the motivation from [18]). However, it is unclear if privacy should really be a concern since there are countless way to track a car besides recording the ID of the TPMS sensors, e.g., the myriad surveillance cameras dispatched on the roads. Besides tracking the user, exposing information related to tire condition to outsiders seems to have no security implication (finally, the condition of tire, in extreme cases, e.g., deflated tire, can be recognized by outsiders from visual inspection). However, if assuring confidentiality of the frame is a desired security objective, then we can enable the encryption option.

The steps perfomed by the sender sensor are summarized in Algorithm 1. The check for authenticity is depicted in Algorithm 2 and proceeds from verifying the session counter and id up to checking the tag which is the more demanding operation (note that verification is performed only by receivers). Finally, Algorithm 3 outlines the steps of the receiver. Here in order to avoid an anonymization of the sensor ID or sending it in cleartext in order to select the particular key, we opted out for trying to decrypt with each key and check its authenticity. This procedure does not add significant additional costs since decryption requires only XOR-ing with a key stream that already exists (and will be used when the frame encrypted with the corresponding key arrives). Moreover, frame verification in case when the selected key is not that of the sender will likely fail at verifying the session id so it does not incur additional costs, e.g., checking for authenticity. To obtain a crisper verification algorithm, since we used encryption in counter mode, one can simply avoid encryption of the ID by using an 8-bit mask. In this case the receiver's algorithm from Algorithm 3 can simply proceed with the key allocated for a particular ID. Encryption of the counter can also raise some concerns in case that frames are lost, however since frame emission takes place at fixed intervals, e.g., 2 minutes, it is easy for the receiver to retrieve the current value of the counter by simply checking the elapsed time since the protocol started. Again, as the counter holds no critical information, it can be sent in plain-text as well (this is just a minor implementation decision).

4.3 Randomness

While strong randomness is vital for the session key initialization, an in-depth analysis of how to generate randomness is out of scope for the current work. Fortunately, there should be enough randomness in the values reported by the sensor, e.g., acceleration in particular, as these depend not only on tire condition, but also on the behavior of the driver. Thus, we can immediately garner whatever output from the sensor data on a randomness pool. We recommend for each sensor to keep a randomness pool that is a simple state variable that XORs over all values that are recorded by the sensor. This state value is used to refresh the authentication key at each renegotiation.

Algorithm 1 Sender's algorithm

```
1: procedure BUILD AND SEND FRAMES
2:     SetSessionKey()
3:     sess_cnt←0
4:     repeat
5:         if sess_cnt>max_sess then
6:             SetSessionKey()
7:         end if
8:         data←ReadSensorData()
9:         frame←concat(data, sess_cnt)
10:        frame←concat(frame, sess_id)
11:        if opt_conf then
12:            kstream←Cipher(K_conf, sess_cnt)
13:            frame←frame⊕ kstream
14:        end if
15:        tag←MAC(K_aut, frame)
16:        frame←concat(frame, tag)
17:        SendFrame(frame)
18:        sess_cnt←sess_cnt+ 1
19:    until true
20: end procedure
```

Algorithm 2 Frame Verification

```
1: procedure CHECKFRAMEAUTHENTICITY(frame)
2:     id←extract(frame, ps_id, ln_id)
3:     sess_cnt←extract(frame, ps_cnt, ln_cnt)
4:     if sess_cnt>sess_cnt[id] then
5:         sess_id←extract(frame, ps_sid, ln_sid)
6:         if sess_id=sess_id[id] then
7:             tag←extract(frame, ps_tag, ln_tag)
8:             tag' ←MAC(K_aut[id], frame)
9:             if tag = tag' then
10:                data←extract(frame, ps_data, ln_data)
11:                return data
12:            end if
13:        end if
14:    end if
15:    return ⊥
16: end procedure
```

Algorithm 3 Receiver's algorithm

```
1: procedure RECEIVE AND VERIFY FRAMES
2:     repeat
3:         frame←ReceiveFrame()
4:         if opt_conf then
5:             for i ← 1, n do
6:                 frame' ←frame⊕ kstream[i]
7:                 data←CheckFrameAuthenticity(frame')
8:                 if data≠⊥ then
9:                     frame←frame'
10:                    exit for
11:                end if
12:            end for
13:        else
14:            data←CheckFrameAuthenticity(frame)
15:        end if
16:        if data≠⊥ then
17:            id←extract(frame, ps_id, ln_id)
18:            sess_cnt[id] ←sess_cnt[id] + 1
19:            kstream[id] ←Cipher(K_conf[id], sess_cnt)
20:        end if
21:        data←⊥
22:    until true
23: end procedure
```

5 Experimental results

Our experimental results are concerned with two aspects: the computational performance of lightweight cryptographic designs and the energy consumption of the protocol, both with respect to the implementation on the SP37 sensor. We discuss these in what follows.

5.1 Lightweight cryptographic designs

The lightweight block ciphers that we chose for our implementation are the SPECK [2] and PRESENT [3] block ciphers. These ciphers were chosen because of their design simplicity which leads to low RAM and flash memory usage. The Katan32 block cipher used by Xu et al. [18] did not appear to be suitable on the SP37 sensor due to higher RAM and Flash memory requirements.

From the lightweight block-cipher family of SIMON and SPECK [2], we choose SPECK as it is designed for optimal performance in software while SIMON is optimal in hardware. SPECK supports a wide range for block and key sizes, starting from 32 bits to 128 bits for the block size, respectively from 64 bits to 256 bits for the key size. We started with an implementation of several SPECK configurations with the code from [7] and [4]. At first, a software configuration larger than SPECK 48/96 was not usable due to RAM limitations. Consequently, we optimized the code in order to minimize RAM usage. This was possible by computing the key stream at the same step where it is used and not by computing and memorizing it in an array for later use (this was the general approach in the available implementations). Thus, the key expansion and the encrypt functions were reduced to a single function. After performing these optimizations, all configurations up to SPECK64/128 could be uploaded on our platform.

The PRESENT block cipher is also suitable for our TPMS application. Although it is optimal in hardware, it is also convenient for software implementations on low-end platforms. From the two available PRESENT configurations, consisting in 64 bit block size and 80 bit vs. 128 bit key, we chose the first one. We started by adapting the available software implementation from [13].

Measurements were done in terms of code size, data size and execution time as can be outlined in Table 2. For memory consumption, the measurements were done with and without the use of compiler optimizations. As can be easily observed, the SPECK block cipher is more suitable for our constrained platform than PRESENT, due to smaller memory needs and execution time.

Block cipher (block/key size)	**Flash(bytes) w/wo opt.**	**Flash(%) w/wo opt.**	**Data(bytes) w/wo opt.**	**Data(% out of RAM) w/wo opt.**	**Duration(ms)**
SPECK32/64*	1032/1371	16,8/22,3	100/117	39/45,7	6,6
SPECK32/64	566/642	9,2/10,4	**0**/21	**0**/8,2	2,5
SPECK48/72	843/1175	13,7/19,1	16/37	6,2/14,4	18,2
SPECK48/96	867/1203	14,1/19,5	20/41	7,8/16	19,3
SPECK64/96	841/1031	13,6/16,7	16/37	6,2/14,4	28
SPECK64/128	865/1059	14/17,2	20/41	7,8/16	29,1
PRESENT64/80	846/1159	13,7/18,8	27/51	10,5/19,9	303,3

Table 2. SPECK and PRESENT performances on Infineon SP37T with and without using compiler optimizations

5.2 Design considerations for energy efficiency

One of the key concerns in implementing the protocol was energy efficiency. To obtain a minimal current consumption, the following key aspects were kept in mind: ensure

minimal consumption of memory resources, ensure protection against battery drain attacks, ensure minimal usage of RF transmitter, parallelize sensor operations, use sensor specific methods for reduced energy consumption (e.g., power-down mode).

For ensuring minimal memory resources, the Speck 32/64 implementation was optimized in terms of RAM and code size. The key expansion step was done together with the encryption step for all results summarized in Table 2 (starting from the 2nd line). Second, the RF data transmission method was chosen considering minimal RAM usage. The dedicated ROM library function for RF transmission was not used because it would require a data buffer where all the RF data is stored before transmission. We did a manual configuration of the transmission procedure, where only the useful data was sent stepwise to the TDA5240.

Ensuring protection against battery drain attacks forced by LF triggers was a significant concern. The default sensor design allows LF telegrams to be received with the following functionalities: requesting sensors measurement, requesting for the sensor ID of a tire, configuring an operation mode or updating the configuration. The last two procedures can lead immediately to battery loss, e.g., forcing the sensor to enter in the diagnose mode will trigger a transmission every 500ms. The first two triggers are not necessarily innocuous as they also lead to waking the sensor up from power-down mode. The driver would not recognize any change or misbehaviour until the battery will run out. As a consequence and since any unnecessary wake-up pattern will lead to battery loss if triggered for sufficiently many times, we decided to ignore LF requests completely. Thus, no wakeup from the power down mode of the 8051 microcontroller and peripherals is generated if an LF telegram is received by the sensor. If these triggers are still needed, a counter must be used to ensure that a maximum number of requests is not overdo. If a maximum value is reached, the TPMS sensor would ignore further requests until the counter reaches the value 0, by decrementing it at every cyclic transmission of the sensor. This would be a feasible solution for the first two kind of requests. For the last two, in case of switching into another operating mode or changing the frequency of transmission of the TPMS sensor, a security measure would be to display this request on the driver dashboard and to wait for his feedback. Of course, all this interaction will be authenticated with the same master key. Due to space constraints, we will not insist in designing sub-protocols for these functionalities since they do not appear to pose additional challenges and do not appear to be compulsory for existing systems.

As stated in the RAM and code size minimization steps, the RF transmitter configuration and transmission was done manually and not by making use of an available ROM library function. This was not the only reason for taking this decision. Another reason is the fact that for achieving minimal current consumption during RF transmission, a control mechanism is needed in order to manage the consumption of the application. This mechanism is provided by the SP37 by a special function register bit, which can be used for indicating that the 8051 microcontroller and other peripherals can enter into an idle mode (a low-current consumption mode) until an event is triggered so that the normal operation can continue. During the RF transmission of one byte this mechanism was used in order to minimize the energy consumption. Starting from the fact that the RF transmission represents the highest current consumption, the values reported by the sensor and the sensor ID was scaled down to 8 bits for the temperature, acceleration

and pressure values with only 7 bits for the sensor ID and 1 bit representing the battery status. This offers of course a baseline for energy consumption and one can increase them at the cost of more energy.

A decision also had to be taken for the measurement of battery voltage. There are two sets of ROM library functions which can be used for this purpose. One is a standalone function (which can be called any-time in the application code) and the other consists in three functions which must be called in a specific order during RF transmission. The last set is recommended in the SP37 ROM library guide [10] because of two optimization aspects: one is the execution time of the two sets and the other one, the parallelization of sensor operations. We opted for the more efficient method.

Besides the optimizations used during the RF transmission, the SP37 sensor has a specific power mode called power-down mode in which only a few peripherals are powered, excluding the microcontroller, which is also waiting for an event in order to wake up and execute the user code. It is important to notice that during most of its lifetime the SP37 sensor is in power down mode.

5.3 Energy consumption

Our measurements included the three cases that can be distinguished: frames without security elements, authenticated frames and encrypted authenticated frames. The difference in energy consumption between the last two is not very high since most of the energy is spent in sending the frame while encrypting the frame does not increase its size. The measurements for standard frames and encrypted authenticated frames are depicted in Figure 5.

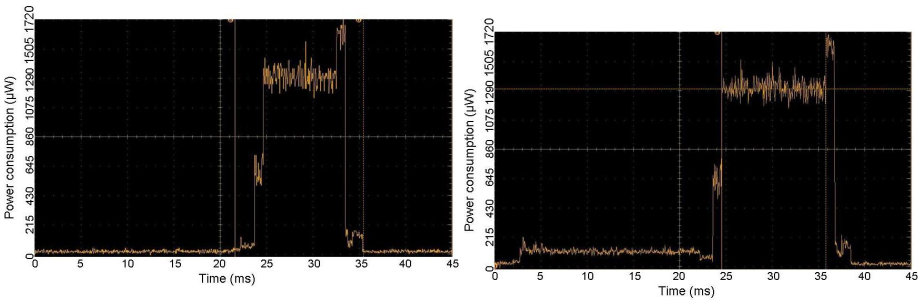

Fig. 5. Power consumption: without (i) and with (ii) encryption and authentication

When comparing the energy consumption for a normal frame with any of the secured frames two differences can be easily observed: the duration of the data processing time and the duration of the RF transmission window. The overall duration required for sending and computing a secured frame is almost double the duration of an unsecured one, 35 ms respectively 29 ms vs. 14 ms. However, in the RF transmission segment (where the highest power consumption levels are reached) the delay generated by the added security bits has a value of 3.4 ms which represents only 22% from the

total frame duration increase and only 9.7% from the total frame duration of a secured frame. Consequently, this addition will not have a big impact on battery discharging. The power consumption during this segment is constant for all frame types at 1.32 mW. Looking at the data processing segment and comparing an unsecured frame with an authenticated one, the delay generated by the authentication has a value of 9.6ms, which represents 64% from the time required to send a regular frame and 33.1% from the time for an authenticated frame. From the power consumption point of view, in case of an unsecured frame we have an average of 92 μW, which is comparable to the average value of an authenticated frame, i.e., 133 μW. Between an authenticated frame and an authenticated-encrypted frame there are minimal differences in term of delays, i.e., 6 ms, due to only two other Speck32/64 calls. The total energy consumption increases from 13.11μJ in case of the standard frames to 20.34μJ in case of the authenticated and encrypted frames. This is roughly by 50% but it is mostly due to increasing the size of the frame with 32 bits for the authentication tag and not due to the cryptographic operations which cost in the order of 2.77μJ. With a larger data frame the added cost will be much lower than 50% as the size of the authentication tag remains constant.

Since in a typical TPMS system the sensor is more than 90% of the time in power down mode, these delays of 3.4ms and 9.6ms will not have a big impact on battery lifetime. By comparing our results to the power consumption measurements from Xu et al. [18], here we have a consumption of only $1.32mW$ during RF transmission, compared to about $450mW$ in [18]. The time delay introduced by processing and sending a secured frame of 13.1ms in their work is comparable with our time delay in the same scope of 14ms (however our platform has lower computational capabilities).

6 Conclusion

Our work shows cryptographic security to be achievable on real-world TPMS sensors at minimal computational costs and energy expenses. For real-world use it is preferable to rely on hardware cryptographic implementations, still, the software implementation that we use in our protocol design, proved to be energy efficient with the additional costs for cryptographic computations being at only 2.77μJ which is around 20% from the total cost of a regular frame transmission at 13.11μJ. This cost can be actually nullified by a proper hardware implementation which is certainly to come for TPMS sensors. At the communication level, the size of the frame is expanded by the 32-bit authentication tag which (along with the costs of the underlying cryptography) increases consumption from 13.11μJ to 20.34μJ which is roughly 50%. We underline that this cost is in fact an upper bound since the authentication tag is equal in size with the actual sensor data (32 bit) while with a larger data-field this percentage decreases as the authentication tag remains constant in size. Compared to the energy consumption reported by previous work [18] our result is two orders of magnitude lower, this is of course mainly due to the dedicated TPMS sensors from the deployment platform but also due to the lighter cryptographic constructions, i.e., SPECK.

Given that a system is only as secure as its weakest link, we consider that securing TPMS interfaces should be taken more serious by automotive manufacturers. Clearly, potential attacks on such systems can now lead to some inconveniences to the drivers

but can have far more serious consequences if TPMS data is going to be used in more critical tasks.

References

1. D. Basin, C. Cremers, and S. Meier. Provably repairing the iso/iec 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.
2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. National Security Agency. THE SIMON AND SPECK FAMILIES OF LIGHTWEIGHT BLOCK CIPHERS. pages 16–45, 2013.
3. A. Bogdanov, R. Knudsen, G. Leander, C. Paar, A. Poschmann, J. Robshaw, and C. Vikkelsoe. PRESENT: AN ULTRA-LIGHTWEIGHT BLOCK CIPHER. IN CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS-CHES 2007. pages 450–466, 2007.
4. 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.
5. 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.
6. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in cryptology-Eurocrypt 2004*, pages 523–540. Springer, 2004.
7. FrenchNationalResearchAgency. SPECK sample source code. http://bloc.project.citi-lab.fr/library.html.
8. Infineon. SP37T 1300kPa.Product brief. http://www.infineon.com/dgdl/SP37_1300kPa_PB.pdf?fileId=db3a3043 382e83730138566f83105c7c.
9. Infineon. *SP37T Datasheet.*, 1.0 edition, January 2010.
10. Infineon. *SP37T ROM Library Guide.*, 1.0 edition, January 2010.
11. Infineon. *TDA5240 Datasheet.*, 4.0 edition, February 2010.
12. R. M. Ishtiaq Roufa, H. Mustafaa, S. O. Travis Taylora, W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium, Washington DC*, pages 11–13, 2010.
13. D. Klose. Ruhr-University-Bochum.PRESENT sample source code. http://www.lightweightcrypto.org/implementations.php.
14. 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.
15. F. Stajano. The resurrecting duckling. In *Security Protocols*, pages 183–194. Springer, 2000.
16. A. Toth. Method and system for monitoring a parameter of a tire of a vehicle, Apr. 16 2014. EP Patent App. EP20,120,464,019.
17. P. van Zyl, S. v. Goethem, S. Jansen, S. Kanarchos, M. Rexeis, S. Hausberger, and R. Smokers. Study on tyre pressure monitoring systems (tpms) as a means to reduce light-commercial and heavy-duty vehicles fuel consumption and co2 emissions. *Final report, European Commission DG Clima*, 2013.
18. M. Xu, W. Xu, J. Walker, and B. Moore. Lightweight secure communication protocols for in-vehicle sensor networks. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, pages 19–30. ACM, 2013.