Fast, Furious and Insecure: Passive Keyless Entry and Start Systems in Modern Supercars

Lennert Wouters[1], Eduard Marin[2,1], Tomer Ashur[1], Benedikt Gierlichs[1] and Bart Preneel[1]

[1] imec-COSIC, KU Leuven Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be
[2] School of Computer Science, University of Birmingham, UK, e.marin@cs.bham.ac.uk

Abstract. The security of immobiliser and Remote Keyless Entry systems has been extensively studied over many years. Passive Keyless Entry and Start systems, which are currently deployed in luxury vehicles, have not received much attention besides relay attacks. In this work we fully reverse engineer a Passive Keyless Entry and Start system and perform a thorough analysis of its security.
Our research reveals several security weaknesses. Specifically, we document the use of an inadequate proprietary cipher using 40-bit keys, the lack of mutual authentication in the challenge-response protocol, no firmware readout protection features enabled and the absence of security partitioning.
In order to validate our findings, we implement a full proof of concept attack allowing us to clone a Tesla Model S key fob in a matter of seconds with low cost commercial off the shelf equipment. Our findings most likely apply to other manufacturers of luxury vehicles including McLaren, Karma and Triumph motorcycles as they all use the same system developed by Pektron.

Keywords: Passive Keyless Entry and Start, reverse engineering, Digital Signature Transponder

Introduction

The first Remote Keyless Entry (RKE) system for cars was introduced by Renault in 1982 [Smi16]. Back then these systems used infrared instead of Radio Frequency (RF) transmissions. Nowadays, vehicles are typically equipped with a RKE system and often include a Passive Keyless Entry (PKE) system. The former allows users to lock and unlock their vehicle by the press of a button on a key fob. The latter unlocks the vehicle automatically when the key fob is in its close proximity.

Additionally, cars employ an immobilizer system that verifies the legitimacy of the user's key fob before starting the engine. According to Van Ours and Vollaard, the use of immobilizer systems reduced the number of car thefts significantly [vOV16]. Immobilizers are usually implemented using a cryptographically enabled Radio-Frequency IDentification (RFID) device embedded in the key fob. The first family of such RFID chips was introduced by Texas Instruments in 1995. These Digital Signature Transponders (DST) relied on the proprietary DST40 cipher [Kai08].[1]

While traditionally keyless entry and immobilization were two separate systems, nowadays luxury cars often combine these features into a single Passive Keyless Entry and Start

[1] RFID chips are sometimes also called RFID transponders and we will use both terms interchangeably.

Licensed under Creative Commons License CC-BY 4.0.

(PKES) system. This allows to unlock the vehicle and disengage the immobilizer without any user interaction. The move towards PKES systems results in higher convenience for users but could also facilitate certain attacks as no interaction of the legitimate user is required. For example, recent reports have shown car thieves using relay attack systems which can be purchased online [FDC11, YZL17, Pol17, AC17].

Initial information gathering

This paper goes beyond relay attacks and assesses the security of the PKES system used in the Tesla Model S. Some initial online searches and the public Federal Communications Commission's (FCC) equipment authorization database revealed that the Tesla Model S key fob uses the Texas Instruments TMS37F128 chip [fcc12]. From the abridged datasheet available at Texas Instruments' website [Tex12c], we expected this system to use the DST80 cipher which was introduced by Texas Instruments in 2008 [Smi16].

Contributions

The main contributions of our work are twofold.

- **Security analysis.** We provide the reader with a detailed explanation of our reverse engineering efforts and protocol analysis. By doing so we identify multiple security issues in the PKES system designed by Pektron such as the use of an inadequate proprietary cipher, the lack of mutual authentication in the challenge-response protocol, no firmware readout protection features enabled and the absence of security partitioning. Additionally, we explore the feasibility of a car-only attack, i.e. without access to the key fob.

- **Proof of Concept attack.** The combination of the identified issues allows for a very efficient attack. We implement a Proof of Concept (PoC) attack allowing us to clone the key fob of high-end vehicles such as the Tesla Model S using Commercial Off The Shelf (COTS) equipment in seconds.[2] We are hence able to unlock and start the vehicle at any time.

Structure

The remainder of this paper is structured as follows. We start by discussing the related work in Section 1. In Section 2 we discover publicly undocumented commands in Texas Instruments DST transponders. In Section 3 we reverse engineer the PKES system followed by a security analysis in Section 4. In Section 5 we implement a full PoC attack on this PKES system allowing us to clone and emulate a target key fob. Additionally, we share our experience with notifying the affected manufacturers in Section 6 and explore potential mitigation strategies in Section 7. Finally, in Section 8 we conclude this manuscript.

1 Related work

Over the years, researchers have shown that manufacturers often rely on proprietary ciphers for their keyless entry and immobilizer systems. The proprietary DST40 cipher was reverse engineered by Bono et al. back in 2005 [BGS+05]. Subsequently, the authors constructed a key cracker using 16 FPGAs allowing the recovery of the 40-bit cryptographic key from two challenge-response pairs in less than one hour. At the time of their research,

[2] We provide a proof of concept video at: https://www.youtube.com/watch?v=aVlYuPzmJoY

DST40 was used in an estimated 150 million vehicle immobilizers and 7 million Speedpass tags for Exxon Mobil's payment system.

The only publicly known attack on DST40's successor DST80 is a physically invasive downgrade attack using a laser and probing station [HT15]. In addition, it requires prolonged physical access to the victim's key fob, expensive equipment and a skilled operator. It is important to note that these researchers had access to the specifications of DST80 during the development of their attack [dst].

Bogdanov was the first to analyze and conduct attacks on the proprietary KeeLoq block cipher owned by Microchip [Bog07]. A more practical algebraic attack was introduced by Indesteege et al. in 2008 [IKD+08]. This attack required 65 minutes of access to the victim's key fob in order to collect the necessary plaintext-ciphertext pairs for recovering the key. Eisenbarth et al. presented a Differential Power Analysis (DPA) attack on a Microchip HCS encoder requiring only ten power traces [EKM+08]. In 2009 Kasper et al. introduced a Simple Power Analysis (SPA) attack on a KeeLoq software implementation [KKMP09]. This led them to recover the manufacturer key from a single power trace.

In 2012 Verdult et al. executed multiple cryptanalytic attacks on the Hitag2 stream cipher used in NXP transponders [VGB12]. The details of this cipher were previously published in [Wie07]. Their attacks allow to recover the 48-bit key in less than 360 seconds. Despite these attacks, Hitag2 is still widely used in vehicle immobilizers and RKE systems. In 2013 Verdult et al. reverse engineered the Megamos Crypto cipher from an automotive key programmer's software. Furthermore, they proposed practical attacks on both the cipher and the authentication protocol [VGE13].

In addition to the weaknesses identified at the cryptographic primitive level, manufacturers often neglect key management. Garcia et al. conducted a security analysis of multiple keyless entry systems used by the Volkswagen Group between 1995 and 2016 [GOKP16]. Their findings revealed that some of these systems rely on a single master key which can be retrieved using commercial programmers. This flaw enables adversaries to clone a key fob after eavesdropping on a single transmission. The authors also proposed a novel attack on a Hitag2-based rolling code scheme used by several car manufacturers. This attack requires eavesdropping on 4–8 transmissions and about one minute of computation. Benadjila et al. analyzed a variant of this Hitag2-based rolling code scheme which is not susceptible to the attack proposed by Garcia et al. They propose an attack through which it is possible to forge valid packets without needing to retrieve the cryptographic key; this attack requires eavesdropping on two legitimate transmissions [BRLK17].

Even if strong cryptographic primitives with unpredictable random keys are used, attacks that exploit weaknesses in the protocol such as relay attacks and eavesdropping-and-jamming may still be possible. The former can be executed by relaying the messages exchanged between genuine devices [FDC11, YZL17]. The latter can compromise the security of rolling-code-based systems by capturing a valid message while jamming it such that the legitimate device is unable to receive it [Kam15, KKMP09]. By repeatedly eavesdropping-and-jamming valid messages and transmitting the previously recorded message, adversaries always have one valid rolling code that will be accepted by the victim's car.

2 DST transponder exploration

The key fobs analyzed during this research use the Texas Instruments TMS37F128 chip. The TMS37F128 is a multi-chip package containing two dies; one identical to the die found in the TMS37126 package and one MSP430F1232 microcontroller. To explore the DST transponders we purchased some TMS37126 chips which can be controlled by an external microcontroller using the Serial Peripheral Interface (SPI) [Tex12b].

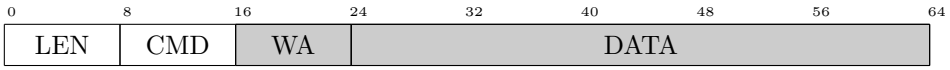

Figure 1: The SPI frame structure. All frames sent to the DST chip include a length (LEN) and command (CMD) field. Some of these frames additionally require write access (WA) and/or data fields. For a specific set of commands, the DST chip will reply with a few data bytes (e.g. in the case of an EEPROM read or response request).

To obtain the datasheets for the TMS37F128 or TMS37126, one has to sign a non-disclosure agreement (NDA) with Texas Instruments which we did not do. Nevertheless, there is one public application note that discloses some basic information about how a microcontroller can interact with the TMS37126 using SPI [MWC12]. For example, it illustrates the SPI frame structure and shows how to interact with the chip to execute DST40.

This section covers the techniques we used to identify publicly undocumented SPI commands. Our initial goal was to learn more about the functionality provided by these transponders and to identify the command used to perform DST80.

2.1 Discovering undocumented SPI commands

We started by designing a simple breakout board for the TMS37126 chip and connected it to an Arduino Pro Mini (3.3 V, 8 MHz) on a breadboard. [3] This was a challenging task due to the vague and sometimes misleading public documentation. For example, one Texas Instruments application note contains two different pinouts for the TMS37126 IC [Tex12a].

This setup allows the Arduino to communicate with the TMS37126 chip using the SPI. In this case the SPI communication requires four connections: (i) clock, (ii) Master Out Slave In (MOSI), (iii) Master In Slave Out (MISO) and (iv) busy. The latter is used by the slave (TMS37126) to signal the master (Arduino) whenever it is ready to receive or transmit the next byte of a frame.

Figure 1 shows the general structure of SPI frames for the DST transponder. All DST SPI frames consist of a length field (LEN) indicating the number of bytes in the SPI frame (excluding the length byte) followed by a command (CMD) field. Additionally, some frames include a write-access (WA) field and a data field.

We found that the behavior of the busy line in combination with two key observations can be used to recover all valid commands supported by the transponder. The first observation is that the busy line produces an error state (i.e. by remaining high or low for an extended period of time) if the CMD of the frame is invalid. Secondly, by setting the value of the LEN byte to `0xFF`, one can determine the correct length for a given command by observing when the busy line indicates an error. For example, if the correct LEN value for a certain command is `0x06`, the busy line will indicate an error after the 6th byte. Using these observations, we were able to uncover all valid values for the CMD and LEN fields.

Applying this technique in an automated fashion revealed four publicly undocumented commands each requiring a 5-byte input (challenge) and returning a 3-byte output (response). Additionally, we found two commands allowing to set two independent 40-bit keys. Table 1 shows an overview of the commands we have identified. After some initial tests with different challenges and keys, we determined that the responses produced by two of these four commands correspond to the ones we would expect from a DST40 implementation. For a given challenge we observed that changing one of the 40-bit keys results in a different response for two of the four commands. From these initial tests, we

[3] We provide a simple schematic with the correct pinout in Appendix B.

Table 1: An overview of the publicly undocumented commands of interest which we discovered. We denote the challenge and keys, respectively, as C and K.

Action	LEN	CMD	WA
DST40(C, K_1)	0x06	0x84	NA
DST_UNK(C, K_1)	0x06	0x85	NA
DST40(C, K_2)	0x06	0x86	NA
DST_UNK(C, K_2)	0x06	0x87	NA
Change K_1	0x07	0x01	0x11
Change K_2	0x07	0x01	0x15

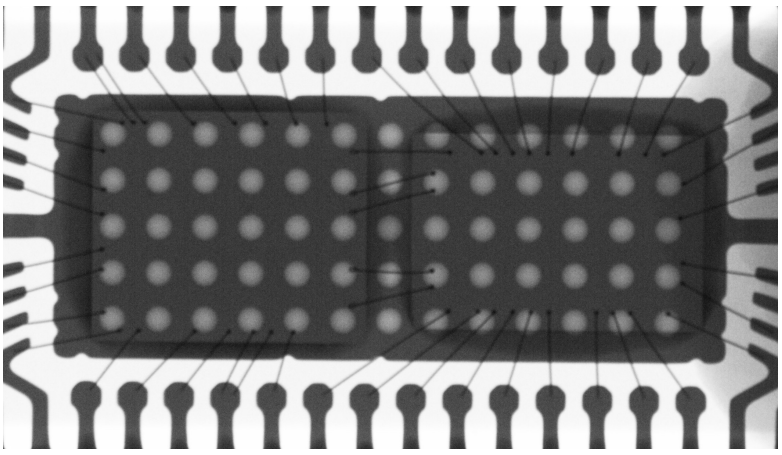

Figure 2: X-ray image of the Texas Instruments TMS37F128. From this image one can identify two dies interconnected by five bond wires. Better viewed on-screen.

were able to conclude that the chip contains two independent DST40 implementations. Additionally, there are two independent implementations of an unknown cryptographic function relying on a 40-bit key. We denote this unknown function as DST40_UNK.[4]

3 Reverse engineering the Tesla Model S PKES system

At this point we had some knowledge about the existing SPI commands. Our next step was to discover how these commands are used in a commercial system. To this end we acquired a Tesla Model S key fob for analysis.

This section describes how we reverse engineered the key fob, the PKES protocol and the rolling code scheme used in the RKE system. We analyze these protocols and detail their security weaknesses. Additionally, we explore the possibility of a car only attack.

3.1 Firmware analysis

There are two ways for recovering the SPI commands used in a key fob with a DST transponder: (i) sniffing the SPI communication or (ii) acquiring and analyzing the microcontroller firmware. Sniffing the SPI communication in the TMS37F128 chip requires connecting to the bond wires interconnecting both dies, as can be seen in Figure 2. Reading the firmware of an MSP430F1232 requires an intact JTAG fuse or circumventing the bootstrap loader (BSL) password check [Goo08].

[4] We also reversed engineered this algorithm and found that it is similar in nature to DST40 but with some minor differences. A description of this algorithm is out-of-scope for this paper and is not included.

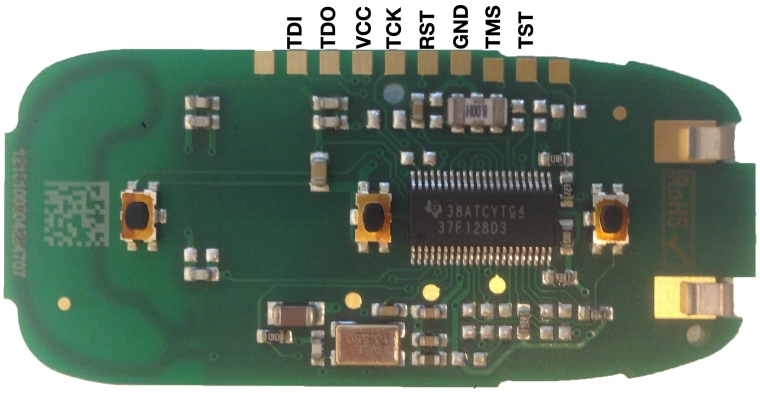

Figure 3: The Tesla Model S key fob PCB with the TMS37F128 chip in the middle. The square shaped pads at the top can be used to connect an MSP430 compatible JTAG debugger.

In the case of the Tesla Model S key fob the security fuse was not blown, Figure 3 shows the key fob's printed circuit board (PCB) with easily accessible debug pads. We used an Olimex MSP430 JTAG debugger [Oli] to read the microcontroller's program memory.

We analysed the retrieved firmware in two stages in order to get a good understanding of its inner workings. In the first stage, we performed static firmware analysis using Binary Ninja [Vec] in combination with the MSP430 plugin [Jos]. In this stage the goal was to get a high level overview of the firmware and to identify the routines to be more closely investigated in stage two. Static analysis involves many hours of reading assembly code, but some information can be taken into account while trying to identify routines of interest [SG16].

The Interrupt Vector Table (IVT), which is stored at the end of the program memory, enabled us to identify the routines that are executed when an interrupt occurs. Using the information from this table we were able to identify the entry point to the firmware image as well as the routines that handle the press of a button on the key fob. Special Function Registers (SFR) can also be used to identify routines of interest. As we are interested in the SPI communication we searched for references to the address of the SPI transmit buffer. Using these techniques we were able to identify a few routines to be analyzed in stage two.

The second stage is dynamic firmware analysis. This analysis involved debugging a real Tesla Model S key fob using the Olimex debugger and the MSPdebug software. The MSP430F1232 supports up to two breakpoints that can be used to halt the processor at certain addresses. When the processor is halted we can inspect the contents of the registers and dump the memory, this is very useful when trying to understand the firmware. For example, by setting breakpoints at the start and end of the SPI routine we could easily identify the bytes being sent and received by the MSP430 microcontroller. In fact, we used this method to identify all the SPI commands that are used during nominal operation of the key fob.

As was already mentioned, we expected the microcontroller to query the transponder for a DST80 response; to our surprise we found that the microcontroller was using only one of the four cipher commands (`0x86`) identified in Section 2.1. Additionally, we had already identified this particular command to be the DST40 compression function.

Using a 40-bit cipher today results in inadequate security levels as even an exhaustive search is feasible in a reasonable amount of time; Bono et al. demonstrated this in 2005 [BGS+05] (see also Section 5.1). While cryptanalysis might reveal weaknesses that further reduce the security of the cipher, it is a time consuming and unnecessary

process for a 40-bit key. In the case of DST40 an exhaustive search will require two challenge-response pairs to recover the correct key as the challenge (40-bit) is larger than the response (24-bit).

In Section 5.1 we show how a vulnerability in the PKES protocol can be exploited by a Time-Memory Trade-Off attack allowing key recovery in a matter of seconds.

3.2 Building a protocol analyzer

In order to understand and analyze the RF protocol used in the PKES system we need to be able to receive and transmit with the same physical layer properties as the car and key fob. The vehicle transmits data over Low Frequency (LF) (134.2 kHz) to the key fob which replies over Ultra High Frequency (UHF) (433.92 MHz). The remainder of this subsection will describe how we built transceivers capable of emulating both the car and key fob.

3.2.1 Low frequency – 134.2 kHz

According to [MWC12] the vehicle uses Burst Width Modulation (BWM). This is also known as Binary Pulse Length Modulation (BPLM) [GPHM10] or Pulse Position Modulation (PPM) [Tex09]. In this modulation scheme bits are encoded by the period during which the carrier is modulated. Bits are separated by a fixed period in which the carrier is unmodulated.

We used a Proxmark III RFID security research tool [GdKGV12] to receive and transmit LF signals, as it supports the 134.2 kHz operating frequency. During this research we had full-time access to a spare Tesla Model S key fob but only limited access to a vehicle. Because of this we first implemented car impersonation, allowing us to use a second Proxmark III to implement receiving.

We configured the Proxmark's relay and multiplexer to use the low frequency peak detect circuit and configured the FPGA to use the peak detection algorithm implemented by iZsh [fai14]. Using one of the microcontroller's timers the number of clock cycles between pulses can be counted in order to demodulate the received signal to bits. While this implementation could successfully receive frames transmitted by a Proxmark it did not allow us to receive frames transmitted by the car.

The peak detect circuit on the Proxmark III is implemented using a diode, causing a voltage drop. This voltage drop in combination with the low received signal strength caused all information to be lost before entering the 8-bit Analog-to-Digital Converter (ADC), making it impossible for the FPGA to detect any peaks. To overcome this issue, we soldered a wire from the output of the low frequency raw input path amplifier to the input of the peak detect circuit. This effectively amplifies the signal before it reaches the analog peak detect circuit. Afterwards we reconfigured the relay and multiplexer in software to accommodate for this change. Additionally, we wrote our own peak detect Verilog code for the FPGA. Our peak detection algorithm allows to more reliably receive transmissions sent by the vehicle over longer distances. However, it is less sophisticated and more application specific compared to the peak detection algorithm implemented by iZsh [fai14]. A complete overview of the implemented demodulation mechanism using the Proxmark III is provided in Appendix C.

3.2.2 Ultra High frequency – 433.92 MHz

We used a software defined radio dongle in order to determine the physical layer parameters such as the operating frequency (433.9 2MHz), symbol rate (2550 symbols/s) and modulation (Amplitude Shift Keying (ASK)). We refer the reader to [Oss16] for more information on the employed techniques.

Using the obtained physical layer parameters we configured a YARD Stick One [Gad] radio transceiver using the RfCat Python library [atl]. This allows us to send and receive data using the same physical layer parameters as the Tesla Model S key fob.

4 Protocol analysis

The analyzed key fobs use two communication protocols. The most frequently used one is the PKES system in which the legitimate key fob automatically engages in a challenge-response protocol with the car (two-way communication). Secondly, the user can lock and unlock the car, open and close the trunk and open the front storage compartment or frunk of the car by the press of a button (one-way communication). In this section we explain how we reverse engineered both protocols used in the Tesla Model S vehicles.

4.1 Passive Keyless Entry and Start

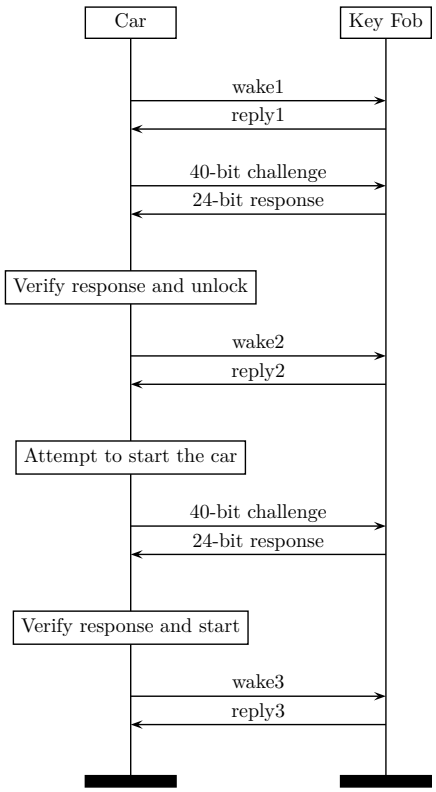

Figure 4: The PKES protocol during nominal operation assuming the key fob is in range of the car.

We created a Python framework handling both the Proxmark III and YARD Stick One, allowing us to receive all communications exchanged by vehicle and key fob during passive entry and passive start.

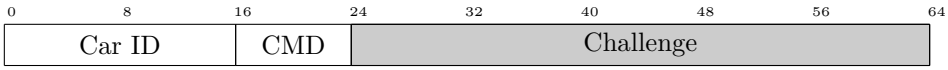

Figure 5: The structure of LF frames transmitted by the car. The 2-byte car ID is followed by a command byte, in case of a response request an additional challenge is concatenated.

Figure 4 gives an overview of the PKES protocol during nominal operation. Our experiments on two different Tesla Model S vehicles show that the general structure of LF frames is as outlined in Figure 5. The car periodically sends one of four 3-byte wake frames consisting of a 2-byte car identifier and a command byte. If the key fob is in range it receives this wake message and replies with a 3-byte frame that is fixed yet unique for each of the four wake messages. Our experiments show that the correct reply to a given wake message can be easily calculated as the latter is an obfuscated version (using byte swaps and logical shifts) of the former.

The car verifies this reply and transmits an 8-byte frame consisting of the 2-byte car identifier, a command byte and a 5-byte challenge. In turn the key fob uses this challenge in the DST40 compression function to produce a 24-bit response before transmitting it over UHF. The vehicle verifies the response and, if successful, unlocks the doors. Afterwards the car and key fob engage in a keep-alive protocol in which the car repeatedly transmits one of the four wake messages to which the key fob replies. Once the driver presses down on the brake pedal to start the vehicle the key fob again authenticates to the vehicle using the same challenge-response protocol described above. This second authentication serves as the immobilizer; it uses the same 40-bit cryptographic key used for opening the car.

As there is no mutual authentication in the protocol the key fob replies to any challenge it receives as long as the frame format is correct. This allows for a chosen input attack which we exploit in our PoC. Additionally, as there is no security partitioning, recovering a single 40-bit key allows an adversary to both unlock the vehicle and start it. Furthermore, the protocol does not implement any distance bounding mechanisms and is thus vulnerable to a relay attack. This was confirmed in practice during a recent study by the Allgemeiner Deutscher Automobil-Club [AC17].

4.2 The remote keyless entry protocol

The RKE protocol allows to lock and unlock the car, open and close the trunk and open the front storage compartment or frunk of the car by the press of a button (one-way communication).

In the previous section we explained how we recovered the protocol and frame structure by effectively building a protocol analyzer. For the RKE protocol we used a different approach in which we did not have to intercept any radio transmissions. For this we used the Olimex debugger in combination with mspdebug and a Saleae logic analyzer [Log] connected to the MICRF112 RF transmitter chip [Mic13] on the key fob PCB. This allowed us to set breakpoints at the start and finish of the SPI routine and retrieve the bytes sent to and received from the DST co-processor.

This technique revealed that following a key press, the key fob will first retrieve a 40-bit counter from the DST transponder's EEPROM. This counter value is used as the input or challenge to the DST40 compression function, using the same 40-bit key as in the PKES system. The final RF packet consists of a preamble followed by an action byte (lock/unlock) concatenated with the DST40 response (3 bytes) and the two least significant bytes of the 40-bit counter. Afterwards the counter is incremented and again stored in the EEPROM. The main goal of the counter is to prevent replay attacks, transmitting part of the counter allows for resynchronization of the car and key fob counters. This kind of construction is often referred to as a rolling code.

This rolling code implementation is vulnerable to a jamming and eavesdropping attack as implemented in Kamkar's RollJam attack [Kam15]. In this attack the adversary uses a transmitter to jam inside the car's receiver bandwidth and a receiver with a narrow bandwidth to capture the key fob's transmissions. The adversary thus ensures that the car is unable to receive the first transmission from the key fob and records it. The same action is performed when the legitimate user uses their key fob a second time; subsequently the adversary transmits the first captured message. At this point the adversary has a legitimate message which is still valid and can be transmitted at a later time. This rolling code scheme is particularly vulnerable to this attack since the action is not authenticated in any way. If the adversary succeeds in obtaining a valid lock command he would be able to unlock the car by changing the action byte before transmission.

4.3 A car-only attack

This section discusses the possibility of a car-only attack. The goal of such attack would be to retrieve the 40-bit cryptographic key without ever being in proximity of the legitimate key fob. This kind of attack could be feasible in practice in a situation were the owner leaves their car for a long period of time (e.g. parking their car at an airport).

The PKES protocol requires the key fob to respond with a valid reply to a wake command transmitted by the vehicle. From our earlier analysis we learned that one can easily predict the correct reply based on the wake message. There is thus no need for an adversary to brute force the correct reply.

The actual attack would go as follows: for each challenge transmitted by the car, the adversary will calculate the response based on a key guess and transmit it. For each challenge transmitted by the car there will be approximately 2^{16} keys which produce a correct response. On average, it would thus take an adversary close to 2^{23} guesses before he finds a key which produces the same response as the key used in the legitimate key fob. Testing 2^{23} keys would take close to 97 days assuming a rate of one guess per second.

After obtaining one challenge-response pair the adversary creates a list of keys which produce the same response for this challenge; this list will contain approximately 2^{16} keys. The last step for the adversary is to test these keys until a match is found. Testing one such key per second would on average result in the correct key within 9 hours.

An adversary would thus be able to recover the 40-bit cryptographic key from a target car without ever being close to the legitimate key fob in less than 100 days on average. It is important to note that this attack can be automated as the car will continue with the keep alive protocol once a correct response is received instead of sending another challenge (indicating that the received response was correct). However, unless an adversary is able to achieve a reasonable speedup (e.g. by increasing the symbol rate, by using de Bruyn sequences or exploiting a flaw in the protocol state machine) it is unlikely to be a practical attack.

A controller area network intrusion detection system could detect this attack if the Body Control Module (BCM) communicates failed authentication attempts. However, in one of our experiments we started the challenge-response protocol and let it time out 40,000 times in a Tesla Model S during a 12-hour period without encountering any issues. Therefore, we believe that no rate limiting is implemented by the vehicle. We note that any naive rate limiting procedure would expose the car to DoS attacks.

5 Proof of Concept implementation

In order to verify our findings we implemented a PoC attack. For this PoC we envision an attack scenario where an adversary is nearby while the legitimate owner parks their car;

the adversary now has a limited amount of time in order to clone the key fob and drive off.

The attack will be carried out in three or four phases:

Phase 0: Receive car wake message (optional). As a first step, the attacker records one wake frame periodically transmitted by the car in order to learn the car identifier. This is an optional task since one could brute force the 2-byte identifier when in proximity of the victim's key fob.

Phase 1: Car impersonation. During this phase, the adversary impersonates the victim's car, transmits two chosen challenges to the victim's key fob, and records the responses. During this phase the adversary would have to be relatively close (roughly 1 m) to the legitimate key fob for a few seconds (e.g. walking by the target) in order to acquire the two challenge-response pairs.

Phase 2: Key recovery. During the second phase the adversary recovers the 40-bit key from these two challenge-response pairs.

Phase 3: Key fob impersonation. After recovering the 40-bit key, the adversary proceeds to mimic the behavior of the victim's key fob to unlock and start the target vehicle.

For Phases 1 and 3 of the proposed attack we made minor modifications to our protocol analyzer. Phase 1 is possible due to the lack of mutual authentication in the protocol. Our implementation of Phase 2 is now explained in detail.

5.1 Key recovery

Bono et al. used 16 FPGAs to brute force the 40-bit key in under 1 hour; today this time could be significantly reduced using more recent FPGAs or FPGA cloud services. Instead of executing a brute force attack using hardware acceleration we opted for a Time-Memory Trade-Off (TMTO) attack [Hel80]. In the case of DST40 there are 2^{16} keys that produce the same 24-bit response to a fixed 40-bit challenge, on average. A second challenge response pair is thus required to determine the correct key. In order to speed up key recovery we grouped all keys producing the same response to a fixed challenge. Resulting in a lookup table consisting of 2^{24} files, each containing roughly 2^{16} keys. The entire file structure takes up 5.4 TB in size. Using this lookup table a key recovery requires two challenge-response pairs. The first pair is used to select the correct candidate key file. The second challenge-response pair is used to recover the correct key via 2^{15} DST40 invocations, on average.

5.2 Practical results

Using our PoC implementation we were able to successfully retrieve the challenge-response pairs from a target key fob in a matter of seconds while being approximately 1 m away. It should be noted that an adversary could attempt to increase this distance, we made no such effort.

Using the first challenge-response pair and the lookup table the key space is reduced to 2^{16} candidate keys. Using the second challenge-response pair we determine the correct key in 2^{15} DST40 invocations, on average. This last step takes less than 2 seconds using a single ARMv7 core running at 1.4 GHz. After recovering the key we are able to successfully unlock and drive away with the vehicle. The entire attack was successfully carried out on two different Tesla Model S vehicles with consent of the owners.[5]

The entire attack platform shown in Figure 6 was implemented using only Commercial Off The Shelf (COTS) equipment. The goal of the PoC attack was not to reduce the cost. Nevertheless, the entire attack platform can be build for 600$ and easily fits in a backpack.

[5]One of these cars was provided by Tesla Motors's engineering team for the demonstration of the attack.



Figure 6: The entire attack platform. Consisting of a Yard Stick one, Proxmark III, Raspberry Pi 3 Model B+, a battery pack and a 6 TB hard drive.

6 Vendor notification

Uncovering vulnerabilities in commercial products always poses a dilemma. On the one hand, announcing the weakness publicly allows third parties to exploit it. On the other hand, hiding it provides no guarantees that it will not be discovered by the less benevolent (or worse, that it already has been discovered and is being exploited). To address this, mechanisms for responsible disclosure of security vulnerabilities have been put into place to ensure proper handling. In what follows, we describe the disclosure procedure we followed.

We first contacted Tesla Motors about these issues on August 31 2017 as per their responsible disclosure guidelines.[6] They provided an initial confirmation of receipt of our email and requested further details on the reported issue. After providing them the requested details, as well as a video of our PoC, they investigated the issue internally and confirmed our findings. At a later stage a Tesla Motors security engineer came to visit us to get a better understanding of the impact of our findings. To that end we successfully demonstrated our attack on one of their engineering vehicles. Additionally, we informed Texas Instruments of our findings related to their TMS37126 and TMS37F128 chips.

In parallel, when we found out that Tesla Motors did not build this PKES system themselves, we contacted both their supplier, Pektron, and other potentially affected manufacturers including McLaren, Karma and Triumph. Initially none of them replied to our emails. After a few months and multiple attempts to make contact we did receive a reply from McLaren and Karma. In both cases we provided the manufacturers with an early version of this paper. Karma thanked us without further questions and McLaren was no longer interested in communicating about the issue once we made clear we would not be removing their name from this paper.

The findings in this paper are already public at the time of submission. We collected the manufacturers' public statements in Appendix A. All manufacturers, except for Pektron and Karma, seem to have made a public statement. While Triumph never replied to any of our emails they did give an official statement to journalists. Their statement acknowledges we contacted them but indicates that they believe our attack to be a relay attack. McLaren's statement indicates that they read our paper as they mention the the-

[6] https://www.tesla.com/about/security?redirect=no

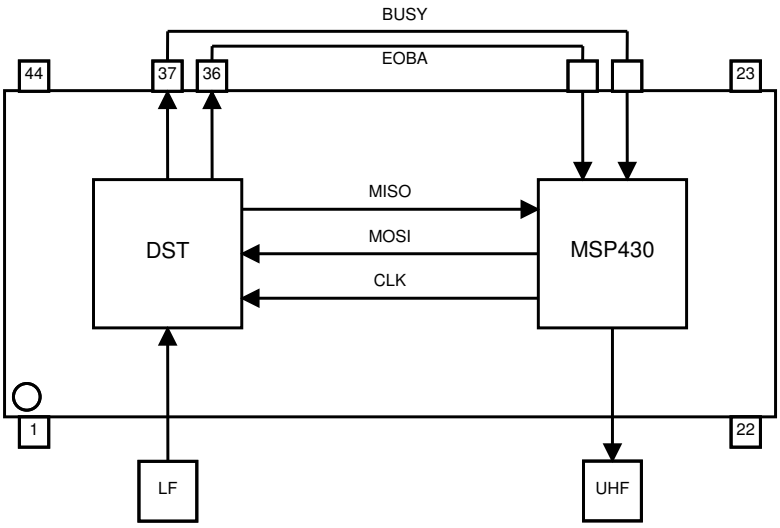

Figure 7: A schematic overview of the TMS37F128 package with the DST co-processor and MSP430 microcontroller. During nominal operation a challenge is receiver over LF and communicated to the MSP430 using the EOBA pin. Afterwards, the microcontroller sends it to the transponder over SPI after which it transmits the returned response over UHF.

oretical car-only attack described in Section 4.3. However, they do not seem to consider the most relevant attack vector.

7 Short term mitigations

In this section we explore potential short-term strategies to mitigate the security weaknesses we have identified. The goal of this section is not to propose a long term solution as this would require extensive redesign of both hardware and software. Due to the lack of information, some of our proposals may not be feasible in practice. In the remainder of this section we assume that the BCM can be updated Over-The-Air (OTA) whereas the key fob will have to be reprogrammed using JTAG or replaced by the manufacturer. Furthermore, as advertised, we assume that the TMS37F128 chips can perform the DST80 operation.

Intuitively, the easiest mitigation strategy would be to upgrade to DST80 since it would require only minor modifications in both the BCM and key fob firmware. If the DST80 cipher indeed offers 80 bits of security, it would preclude exhaustive search and TMTO attacks for opponents with moderate budgets. While this solution would not require any changes to the key fob hardware it does rely on the proprietary DST80 cipher which has not received any public scrutiny. This is therefore only a short term solution, no new vehicle or other product should rely on DST80 for security.

As DST80 is a proprietary cipher it would be worth exploring the possibility of implementing a lightweight, secure cryptographic cipher in the MSP430 firmware. In such case, the DST chip would be used only for its LF analog front-end and EEPROM storage. This is possible since the DST chip receives the challenge transmitted by the car and sends it to the MSP430 microcontroller using the EOBA pin, as shown in Figure 7. Afterwards, the microcontroller sends the challenge back to the DST chip over SPI in order to get a DST40 response which is transmitted over UHF. However, this approach would require careful implementation and a certain expertise to not introduce new vulnerabilities.

In addition to upgrading the cipher suite, the communication protocols should be

updated. For example, the challenge-response protocol should include mutual authentication. According to the public information provided by Texas Instruments [Tex12c], mutual authentication is supported.

The RKE protocol should be modified to authenticate the desired action in order to make jamming and eavesdropping attacks harder to execute in practice. Additionally, it would be desirable to introduce some partitioning in the system. If the DST transponder allows to set multiple 80-bit keys the system should use independent random cryptographic keys for unlocking and starting the vehicle.

Public information shows that Tesla introduced an OTA update that allows users to disable the passive entry functionality and to enable Pin to Drive [Gre18]. Additionally, all Tesla Model S vehicles produced from June 2018 onward use a newer key fob, existing Tesla Model S owners can purchase the upgraded key fobs.

8 Conclusion

We analyzed the PKES system used in some of the most advanced cars on the market. Our evaluation revealed that this system relies on a 40-bit proprietary cipher to unlock and start the vehicle, and reuses the same 40-bit key for both functions. Therefore, there is no security partitioning in this system. In addition to being vulnerable to relay attacks, the protocol does not implement mutual-authentication enabling us to execute a chosen input attack. The presented PoC attack allows us to clone a target key fob in a matter of seconds using low cost equipment. Additionally, we discovered publicly undocumented commands in the chips used by this system which are likely used for the proprietary DST80 cipher.

Despite extensive effort from the security research community to show the insecurities associated with proprietary ciphers they are still used today. DST40 was first shown to be insecure in 2005, yet fourteen years later it is still being used in high-end vehicles. We would like to emphasize that there is no good reason to use proprietary ciphers over well-scrutinized cryptographic primitives. For example, Texas Instruments offers similar products using the AES [Ins13].

Furthermore, the secrecy of datasheets only hinders adversaries, security researchers and professionals and does not provide real security. The fact that these datasheets are secret makes it more difficult for any car manufacturer to conduct a thorough review of the system they purchased from a third party. One can only wonder which additional vulnerabilities could be revealed if these datasheets were ever leaked.

Additionally, we would like to encourage vehicle manufacturers to work more closely with their supplier in order to prevent these issues in the future. Moreover, any company that produces security and safety critical products should have a publicly documented procedure for security researchers to report vulnerabilities.

Acknowledgments

The authors would like to thank Johan Vanhulst from the KU Leuven Department of Materials Engineering for helping us with making the X-ray images, and Bert Deknuydt for maintaining the computation pool at the KU Leuven Department of Electrical Engineering. Furthermore we would like to thank Tesla Motors for constructive discussions during the proper responsible disclosure procedure, and Amit Elazari as well as our university's legal department for assisting with legal and ethical disclosure questions. Finally, we would like to thank the car owners who gave us permission to perform tests on their vehicles.

This work was supported in part by the Research Council KU Leuven: C16/15/058, the Flemish Government through FWO Sancus G.0876.14N, by the Hercules Foundation AKUL/11/19, through the Horizon 2020 research and innovation programme un-

der grant agreement Cathedral ERC Advanced Grant 695305, under grant agreement H2020-FETFLAG-2018-03-820405 QRANGE, under grant agreement H2020-DS-LEIT-2017-780108 FENTEC and the NIST project.

References

[AC17] ADAC Allgemeiner Deutscher Automobil-Club. Weiterhin Sicherheitslücke bei Komfortschlüsseln - Fahrzeuge mit Keyless leichter zu klauen. https://www.adac.de/infotestrat/technik-und-zubehoer/fahrerassistenzsysteme/keyless/, 2017. [Online; accessed 06-February-2018].

[atl] atlas0fd00m. rfcat. https://github.com/atlas0fd00m/rfcat. Version: 65535.

[BGS+05] Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D Rubin, and Michael Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. In *USENIX Security Symposium*, volume 31, pages 1–16, 2005.

[Bog07] Andrey Bogdanov. Linear Slide Attacks on the KeeLoq Block Cipher. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Xining, China, August 31 - September 5, 2007, Revised Selected Papers*, volume 4990 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2007.

[BRLK17] Ryad Benadjila, Mathieu Renard, José Lopes-Esteves, and Chaouki Kasmi. One Car, Two Frames: Attacks on Hitag-2 Remote Keyless Entry Systems Revisited. In William Enck and Collin Mulliner, editors, *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017*. USENIX Association, 2017.

[Dow18] Alan Dowds. Triumph issues advice to prevent keyless ignition hacking. https://www.visordown.com/news/general/triumph-issues-advice-prevent-keyless-ignition-hacking, 2018. [Online; accessed 21-Nov-2018].

[dst] Private communications with one of the authors.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2008.

[fai14] RFID hacking preamble: a new peak detection for the proxmark3. https://fail0verflow.com/blog/2014/proxmark3-fpga-peak-detection/, 2014. [Online; accessed 7-May-2018].

[fcc12] OET Exhibits List for FCC ID AQO002. fcc.io/AQO002, 2012. [Online; accessed 7-May-2018].

[FDC11] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.

[Gad] Great Scott Gadgets. YARD Stick One - a sub-1 GHz wireless test tool controlled by your computer. https://greatscottgadgets.com/yardstickone/. [Online; accessed 7-May-2018].

[GdKGV12] Flavio D Garcia, Gerhard de Koning Gans, and Roel Verdult. Tutorial: Proxmark, the swiss army knife for RFID security research. *Technical Report, Radboud University Nijmegen*, 2012.

[GOKP16] Flavio D. Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. Lock It and Still Lose It - on the (In)Security of Automotive Remote Keyless Entry Systems. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 929–944. USENIX Association, 2016.

[Goo08] Travis Goodspeed. Practical attacks against the msp430 bsl. In *Twenty-Fifth Chaos Communications Congress. Berlin, Germany*, 2008.

[GPHM10] Jim Goings, Toby Prescott, Michael Hahnen, and Karl Militzer. *Design and Security Considerations for Passive Immobilizer Systems*, 2010.

[Gre18] Andy Greenberg. Hackers can steal a tesla model s in seconds by clonign its key fob. https://www.wired.com/story/hackers-steal-tesla-model-s-seconds-key-fob/, 2018. [Online; accessed 11-Sep-2018].

[Hel80] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, 26(4):401–406, 1980.

[HT15] David Hulton and Christopher Tarnovsky. Gone in 60 Seconds. https://prezi.com/5gxalfdj9eyz/copy-of-gone-in-60-seconds/, 2015. [Online; accessed 7-May-2018].

[IKD+08] Sebastiaan Indesteege, Nathan Keller, Orr Dunkelman, Eli Biham, and Bart Preneel. A Practical Attack on KeeLoq. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[Ins13] Texas Instruments. Car access product family. http://www.ti.com/lit/ml/slyt455a/slyt455a.pdf, 2013. [Online; accessed 7-May-2018].

[Jos] Josh Watson. MSP430 Architecture plugin for Binary Ninja. https://github.com/joshwatson/binaryninja-msp430. Version: 0.2b.

[Kai08] Ulrich Kaiser. Digital Signature Transponder. In Paris Kitsos and Yan Zhang, editors, *RFID Security: Techniques, Protocols and System-on-Chip Design*, pages 177–189. Springer US, Boston, MA, 2008.

[Kam15] Samy Kamkar. RollJam. https://samy.pl/defcon2015/, 2015. [Online; accessed 7-May-2018].

[KKMP09] Markus Kasper, Timo Kasper, Amir Moradi, and Christof Paar. Breaking KeeLoq in a Flash: On Extracting Keys at Lightning Speed. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International*

Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings, volume 5580 of *Lecture Notes in Computer Science*, pages 403–420. Springer, 2009.

[Log] Saleae Logic. Saleae Logic. The logic analyzer you'll love to use. https://www.saleae.com/. [Online; accessed 7-May-2018].

[McL18] McLaren. Potential vehicle security risk. https://cars.mclaren.com/ownership/recalls-1, 2018. [Online; accessed 21-Nov-2018].

[Mic13] Microchip. *MICRF112 - 300MHz to 450MHz, +10dBm, 1.8V to 3.6V, ASK/FSK Transmitter with Shutdown*, 2013.

[MWC12] Herbert Meier, Erik Wallander, and Jim Childers. *Passive Go Transponder With Novel Bi-Directional Low Frequency Communications*, 2012.

[Oli] Olimex. MSP430-JTAG-TINY-V2. https://www.olimex.com/Products/MSP430/JTAG/MSP430-JTAG-TINY-V2/. [Online; accessed 7-May-2018].

[Oss16] Michael Ossmann. Rapid Radio Reversing. *Black Hat Briefings*, 2016.

[Pol17] West Midlands Police. Police release footage of 'relay crime'. https://www.west-midlands.police.uk/news/4544/watch-police-release-footage-relay-crime, 2017. [Online; accessed 7-May-2018].

[SG16] Ryan Speers and Travis Goodspeed. A Tourists Phrasebook for Reversing MSP430. *International Journal of PoC or GTFO*, 0x11:28–32, 2016.

[Smi16] Craig Smith. *The Car Hacker's Handbook: A Guide for the Penetration Tester*. No Starch Press, 2016.

[Tex09] Texas Instruments. *Passive low frequency interface device with EEPROM and 134.2 kHz transponder interface*, 2009.

[Tex12a] Texas Instruments. *3D Analog Front End: Migrating From TMS37122 and TMS37127 to TMS37126D3*, 2012.

[Tex12b] Texas Instruments. *TMS37126 - Remote access identification device with integrated 3D wakeup receiver and immobilizer interface*, 2012.

[Tex12c] Texas Instruments. *TMS37F128 - Controller Remote Access Identification (CRAID) with integrated microcontroller, 3D wakeup receiver, and immobilizer interface*, 2012.

[Vec] Vector 35. Binary Ninja. https://binary.ninja/. Version: 1.0.13.

[VGB12] Roel Verdult, Flavio D. Garcia, and Josep Balasch. Gone in 360 Seconds: Hijacking with Hitag2. In Tadayoshi Kohno, editor, *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 237–252. USENIX Association, 2012.

[VGE13] Roel Verdult, Flavio D Garcia, and Baris Ege. Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer. In *USENIX Security Symposium*, pages 703–718, 2013.

[vOV16] Jan C. van Ours and Ben Vollaard. The Engine Immobiliser: A Non-starter for Car Thieves. *The Economic Journal*, 126(593):1264–1291, 2016.

[Wie07] I.C. Wiener. Philips/NXP Hitag2 PCF7936/46/47/52 stream cipher reference implementation. http://cryptolib.com/ciphers/hitag2, 2007.

[YZL17] Qing Yang Yingtao Zeng and Jun Li. Chasing Cars: Keyless Entry System Attacks. https://www.youtube.com/watch?v=rhm1TiFJc7s, 2017. [Online; accessed 7-May-2018].

Appendices

A Vendor responses

A.1 Tesla Motors

"Due to the growing number of methods that can be used to steal many kinds of cars with passive entry systems, not just Teslas, we've rolled out a number of security enhancements to help our customers decrease the likelihood of unauthorized use of their vehicles.

None of these options would be possible for any traditional automaker our ability to update software over the air to improve functionality and security is unique.

Based on the research presented by this group, we worked with our supplier to make our key fobs more secure by introducing more robust cryptography for Model S in June 2018. A corresponding software update for all Model S vehicles allows customers with cars built prior to June to switch to the new key fobs if they wish.

In addition, we had already been working on several other over-the-air updates to help protect our customers from thefts last year we introduced an update that allows all customers to turn off passive entry entirely, and this year we introduced PIN to Drive, which allows customers to set a unique PIN that needs to be entered before their vehicle is driven." [Gre18]

A.2 McLaren Automotive

"We have been alerted to a potential new security risk that may affect McLaren as well as the products of several other luxury automotive brands.

The relay attack that has previously been reported in the media suggested that a car key could be scanned if left within range of the vehicle and the car driven away. The car, however, cannot be restarted as the key was still with its owner.

A second potential method involves the vehicle being scanned by custom equipment left in close-proximity and decoded in 100 days. A cloned key can be produced as above.

To reassure you, the method is currently considered as a low-risk by experts because the custom equipment required for such a breach, while feasible, requires skilled assembly and use. The equipment also needs to be in very close proximity to a key. To further reassure you, the vulnerability has not been proven to affect our vehicles and we know of no McLaren that has been compromised in such a way.

Nevertheless, we take the security of all McLaren vehicles extremely seriously. Our in-house experts are therefore already working closely with suppliers and

the industry to investigate further. The best method to currently protect your McLarens key from malicious close-proximity scanning is to keep it within a signal blocking pouch while not in use.

Over the weekend we started contacting every owner of a new or pre-owned McLaren to inform them of the issue and to offer them a signal blocking pouch free-of-charge. This is the only method currently proven to prevent close-proximity scanning. Please note, however, that it does not help prevent the 100 day cloning method mentioned above. For this, software changes to the vehicle are required which are being investigated.

We are teaming up with a supplier to secure enough McLaren-standard pouches to meet the needs of all our owners which will be made available free-of-charge via their McLaren retailer over the next few weeks.

If you have any further questions, please contact McLaren Client Services by emailing client.services@mclaren.com or by calling +44 (0) 1483 261500.

While this potential method has not been proven to affect our cars and is considered to be a low-risk, plus we have no knowledge of any McLaren vehicle being stolen by this or the previously reported relay attack method, nevertheless we take the security of our vehicles and the concerns of our customers extremely seriously. Therefore we have already begun to write to every owner of a new or pre-owned McLaren (including 4000 customers across the US) for whom we have contact information to alert them to the risk. The communication reassures them that our in-house experts are working closely with suppliers and the industry to investigate further, plus we offering each owner a Signal Blocking Pouch at no cost." [McL18]

A.3 Triumph Motorcycles

"Triumph have previously been contacted by the researchers at KU Leuven with regards to their study and are, of course, conscious of cyber theft.

This type of attack has received a reasonable amount of press coverage in the UK, specifically for vehicle (car) theft, where the key fob has an operating distance and will continue to communicate with the vehicle if within that vicinity. If the key is out of reach or switched off there is no signal to intercept. The Triumph key fob includes an additional layer of security allowing the rider to disable that signal manually, no matter the proximity to the motorcycle, and as such significantly reduces the risk of theft when utilised, something that is recommended as part of the handover procedure.

We continue to innovate with technology on motorcycles, whether that be security, safety or functionality and therefore continue to monitor any attempt to overcome these features within our research and development." [Dow18]

B Example schematic

The schematic in Figure 8 allows one to connect a TMS37126 IC to an Arduino Pro Mini (3.3 V, 8 MHz). D2, D5, D11, D12 and D13 refer to digital pins on the Arduino board.

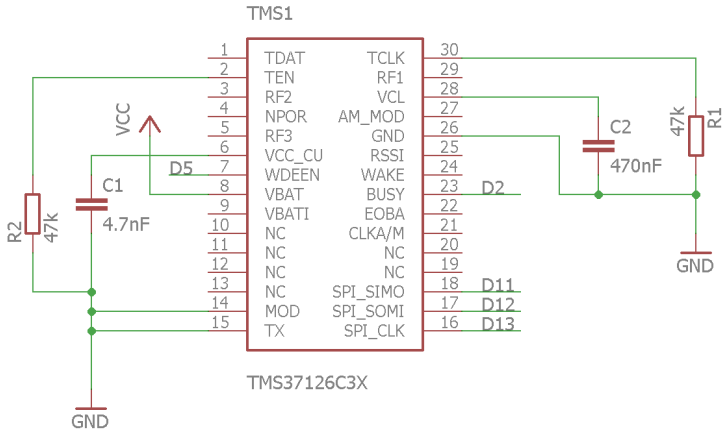

Figure 8: The correct pinout for the TMS37126 ICs we tested and an example schematic of how to connect it to an Arduino Pro Mini.

C Receiving car transmissions using the Proxmark III

Figure 9 shows a LF frame being captured by the Proxmark's antenna and the different stages it goes through before being demodulated by the microcontroller.

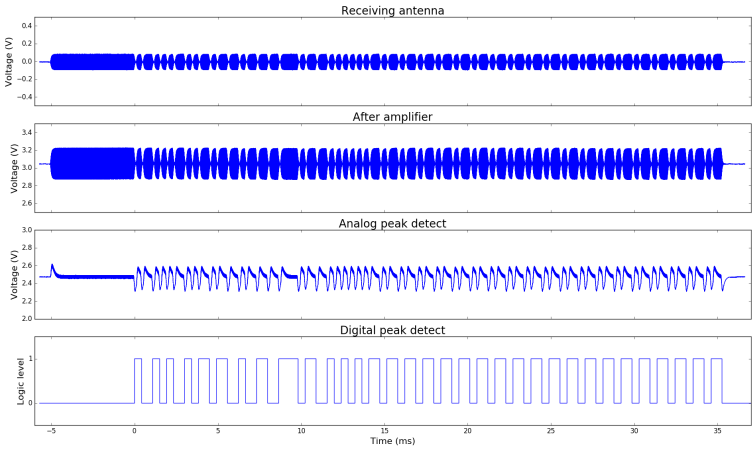

Figure 9: The first (top) plot shows the analog signal received by the antenna (measured at TP2). The second plot shows the analog signal after passing through the Proxmark's LORAW receive path (measured at C41). The third plot shows the result of the Proxmark's analog peak detect circuit (measured at TP1), this analog signal is digitized by an 8-bit ADC, the output of which is processed by the FPGA to produce the final digital peak detect signal (measured at TP7). This signal is processed by the microcontroller and results in the demodulated signal.