UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Daniel Würsch

Security Analysis of Skybrake DD5 immobilizer

Master's Thesis (30 ECTS)

Supervisor: Arnis Paršovs, PhD

Tartu 2024

Abstract

Security Analysis of Skybrake DD5 immobilizer

Car immobilizers are devices intended to prevent car theft by immobilizing the car in absence of a unique authentication token held by the authorized driver of the car. This work documents the process of performing a security analysis of the Skybrake DD5 immobilizer in order to understand its working principles and to verify the security claims by the manufacturer. The main working principles of the immobilizer have been reverse engineered and methods of capturing and injecting signals using commonly sold and available hardware have been documented. By analysis of captured signals from the immobilizer, the authentication protocol flow and the content of the exchanged messages have been reverse engineered and documented. Possible attack vectors were described and analyzed in order to assess if the immobilizer is vulnerable to them. While this work does identify some minor weaknesses and areas of concerns, no major vulnerabilities were found during the security analysis.

Keywords: immobilizer, reverse engineering, security analysis

CERCS: T120 Systems engineering, computer technology; T121 Signal processing; T170 Electronics

Skybrake DD5 auto immobilaiseri turvaanalüüs

Autode immobilaiserid on seadmed, mis on ette nähtud autovarguste ennetamiseks immobiliseerides auto, kui volitatud juhti, kes tavaliselt kannab endaga ainulaadset autentimiskaarti, ei ole läheduses. Käesolev töö dokumenteerib Skybrake DD5 immobilaiseri turvaanalüüsi tegemise protsesse, et mõista selle tööpõhimõtteid ja kontrollida tootjapoolseid turvanõudeid. Immobilaiseri peamised tööpõhimõtted on pöördkonstrueeritud ning vabalt müüdava ja saadaoleva riistvara abil liikunud andmed on dokumenteeritud. Pöördkonstrueeritud ja dokumenteeritud on immobilaiserist püütud signaalide analüüsimisel autentimisprotokolli voog ja vahetatud sõnumite sisu. Võimalikud rünnakuvektorid dokumenteeriti ja vaadati üle, et hinnata, kas immobilaiser on nende suhtes haavatav. Kuigi see töö tuvastab ka mõned väikesed turvanõrkused ja murekohad, ei leitud turvaanalüüsi käigus suuri puudusi.

Võtmesõnad: immobilaiser, pöördkonstrueerimine, turvaanalüüs

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia; T121 Signaalitöötlus; T170 Elektroonika

Contents

List of Tables

List of Figures

Acronyms

AES	Advanced Encryption Standard
ARM	Advanced RISC Machine
CLK	Clock
CRC	Cyclic Redundancy Check
CRLF	Carriage Return and Line Feed (line termination)
CS	Chip Select
CU	Control Unit
DPL	Dynamic Payload Length
ECB	Electronic Code Book (Block cipher mode of operation)
EEPROM	Electrically Erasable Programmable Read-Only Memory
EU	European Union
GFSK	Gaussian Frequency Shift Keying
GND	Ground
ICSPCLK	ICSP Clock signal
ICSPDAT	ICSP Data signal
ICSP	In-Ciruit Serial Programming
IPE	Integrated Programming Environment
LF	Line Feed (line termination)
MCLR	Master Clear signal
MISO	Master In Slave Out
MOSI	Master Out Slave In
NC	Normally Closed
PCB	Printed Circuit Board
PCF	Packet Control Field
PIN	Personal Identification Number
PT	Personal Transceiver
RAM	Random-Access Memory
RX	Receive
SDR	Software Defined Radio
SID	Used vendor to refer to randomly generated challenge
SN	Serial Number
SOC	System On Chip
SPI	Serial Peripheral Interface
SRAM	Static Aandom-Access Memory
TTL	Transistor-Transistor Logic
TX	Transmit
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

In order to avoid any misunderstandings when dealing with numbers using different bases, all numbers which are not in decimal have their base indicated as subscript, either *b* for binary (base 2) or *h* for hexadecimal (base 16). Taking the decimal number 42 as an example, the same number might be represent in this thesis as $\texttt{2A}_\text{h}$ in hexadecimal or $\texttt{00101010}_\text{b}$ in binary.

1 Introduction

Car immobilizers are electronic devices which aim to enhance the security of vehicles by preventing car theft. They consist of the immobilizing unit installed within the car and a transmitter held by the authorized driver of the car.

If the engine ignition is attempted without the presence of the authorized transmitter, the immobilizing unit within the car will prevent the engine from starting or disable the engine shortly afterwards. This creates an additional security beyond standard mechanical locking mechanisms to improve security and deter potential thieves who may not have access to the immobilizer transmitter.

Many countries, including members of the European Union [1], have regulations which make it mandatory for new cars to be equipped with factory-fitted immobilizers by the car manufacturer. As a result of those regulations, many modern cars sold around the world have factory-fitted immobilizers provided by the manufacturer of the car.

In addition to factory-fitted immobilizers, there are third party after-market immobilizers which can be installed in cars. After-market immobilizers are popular for cars without factory-fitted immobilizer, or whose factory-fitted immobilizer is no longer considered secure. Insurance companies drive the adoption of third-party car immobilizers, as insurance policies often require the policyholder to install additional security systems in order to be insured against car theft [2] [3].

1.1 Problem description

Car theft can result in large financial loss and car immobilizers aim to deter car thieves and prevent car thefts. Therefore it is crucial that those devices fulfil their promise and cannot be easily bypassed by car thieves. Manufacturers of car immobilizers are hesitant in providing detailed documentation about the working principle of their product, and there often is only marketing material available for end customers assuring its security to the customer. Keeping the working principles a secret creates hurdles for an in-depth security analysis and customers cannot verify if the manufacturer delivers on their security promises. There are known cases of car immobilizers which have been shown to be insecure [4], using either broken cryptographic primitives, or using *security through obscurity* to hide flawed and vulnerable protocols.

1.2 Aim of this work

In this thesis, a security analysis of an after-market car immobilizer Skybrake DD5 manufactured by a Latvian manufacturer Autonams will be performed. In 2013, Laks performed an in-depth security analysis [5] of the predecessor Skybrake DD2. In his work, he discovered multiple vulnerabilities and the encryption scheme itself turned out to be insecure, relying only on secrets which were shared across all devices.

The goal of this thesis is to provide insights into the working principles and resulting security guarantees of the successor model Skybrake DD5 and highlight discovered findings, weaknesses and attack vectors. The main goals of this thesis are:

1. Discovering what data is being transmitted by the car immobilizer
2. Discovering how the car immobilizer authenticates the personal transceiver
3. Exploring attack vectors like spoofing signals or injecting malicious signals

In order to find answers to the questions listed above, the following methods have been used:

1. All the information publicly available about the immobilizer or its components has been analyzed. This information consists mainly of user manuals, installation guides and datasheets of integrated circuit components.
2. The information was verified by the author of this work in a lab environment using two available immobilizers which he had access to.
3. The printed circuit board (PCB) of the components were probed for interesting and relevant signal traces. The PCBs of the components were modified by adding additional connectors exposing signals of interest.
4. A custom debugging tool was written which allows capturing and transmitting radio packets compatible with the immobilizer.
5. Immobilizer authentication protocol messages were captured over longer time periods, and statistical analysis was performed of the recorded data.
6. Identified attack scenarios were implemented in the custom debugging tool and tested against the available immobilizers.
7. Methods for dumping the memory of the immobilizer's microcontroller have been explored and attempted.

1.3 Structure of the thesis

The main body of the thesis starting at Chapter 2 will introduce the Skybrake DD5 immobilizer based on publicly available materials and manuals. The following Chapter 3 will delve into information that can be obtained by passively observing the immobilizer and its signals during its operations to obtain first clues of the working principles of the immobilizer. Chapter 4 explores various attack vectors and the immobilizer's behavior will be further explored by repeating signals and injecting modified signals. In Chapter 5, methods for gaining access to the memory of the microcontrollers are explored.

2 Skybrake DD5 immobilizer

Skybrake DD5 is an after-market immobilizer which is produced and distributed by the Latvian manufacturer Autonams. It was first mentioned in July 2013 [6] and is still a supported and listed product by the manufacturer as of August 2024 [7].

This chapter introduces the Skybrake DD5 immobilizer and describes its main functionalities, features and security claims.

2.1 Skybrake DD5 package contents

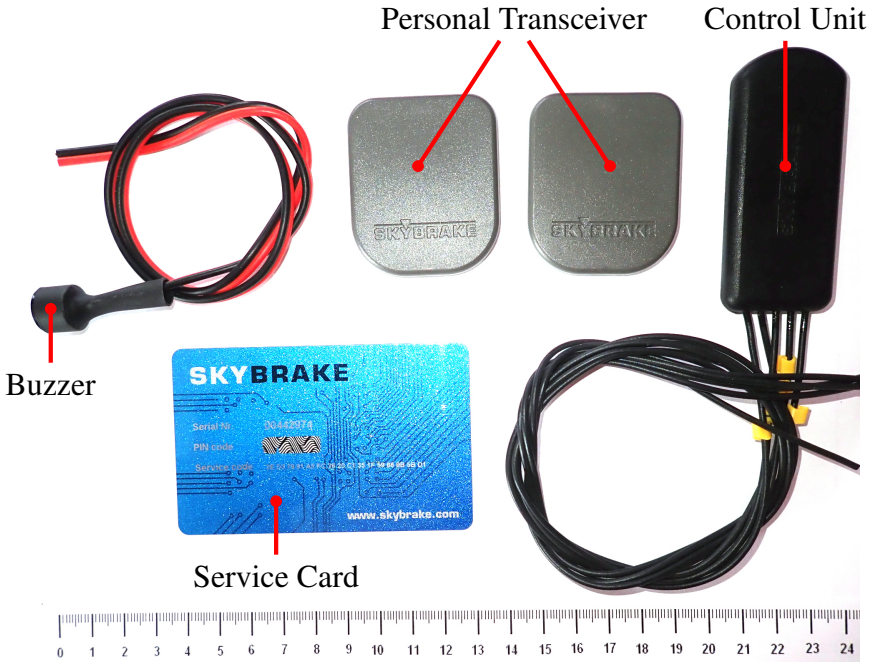


Figure 1: Contents of the Skybrake DD5 cardboard box

The Skybrake DD5 immobilizer is sold in a cardboard box. The content of the cardboard box is shown in Figure 1. The cardboard box contains the control unit (CU), buzzer and two personal transceivers (PT). Additionally, a service card is included with the immobilizer's serial number (SN), the PIN code and a service code.

The author of this work had access to two Skybrake DD5 car immobilizer sets. Table 1 shows the information that was obtained from the cardboard box and the service cards of the immobilizers. The user manual [8] mentions that model 521100 is equipped with a motion detector, whereas models 520100 and 520200 are not equipped with motion detector sensors. Unfortunately no public information could be found highlighting the

features and differences between the models 520120 and 520160 which were available to the author of this work.

Model	Serial Nr.	Service Code
520120	442974_{h}	$\texttt{2E D3 79 91 A5 FC 26 20 C1 35 1F 59 86 9B 5B D1}_{\text{h}}$
520160	$4457\texttt{FC}_{\text{h}}$	$\texttt{B0 A1 1A 10 AB 15 A1 89 77 03 F1 AC 62 B0 51 46}_{\text{h}}$

Table 1: Analyzed Skybrake DD5 car immobilizer sets

2.2 Basic working principle

Once the car ignition is switched on, the control unit enters the activation state. While in activation state, the car is not immobilized and the engine of the car can be started. If no personal transceiver is in the control unit's vicinity and 20 seconds have passed, the control unit enters into a warning state. While in warning state, the control unit will engage the buzzer to alert the driver about the missing personal transceiver. After an additional 10 seconds, it will immobilize the car by turning off the engine.

The control unit will enter into the authenticated state if one of the personal transceivers is in its vicinity within approximately 5 m of the control unit [8]. The car will not be immobilized while the personal transceiver is in its vicinity and remain in authenticated state, thus the engine will continue to run or can be started if it has been previously immobilized.

2.3 Features of the car immobilizer

This section will highlight the main features of the car immobilizer which are advertised in its user manual [8].

Anti-Hijack Function: This prevents the car from being hijacked after the engine has been started. The control unit monitors for the presence of the authorized personal transceiver throughout the travel duration. The control unit will enter a warning state if the personal transceiver is missing for more than 20 seconds. While in warning state, the immobilizer engages the buzzer to inform the driver about the imminent engine shutdown. After an additional 2 minutes, the control unit will immobilize the car by turning off the engine. There are safety implications when the engine is disabled during driving, and the user manual [8] prohibits this feature from being enabled within the European Union and Russian Federation.

Low Battery Warning: The personal transceiver transmits the battery status to the control unit. In case of low battery, the control unit will sound the buzzer every

time the ignition of the car is switched on to inform the driver that the battery of the personal transceiver needs to be changed.

Motion Detection: Some Skybrake DD5 models are equipped with motion detection sensors. Immobilizers equipped with motion detection sensors will not immobilize the car unless it starts moving. This allows the user to leave the car switched on during cold winters or hot summers for heating or cooling. None of the models available to the author of this work have been equipped with motion detection.

Emergency PIN Code: The PIN code from the service card allows to temporarily disable the car immobilizer. This allows the card holder to still drive the vehicle when the personal transceiver is lost or malfunctioning. The PIN will be entered by turning the car engine ignition on and off in a particular sequence, as the car engine ignition is the only input mechanism of how the user can communicate with the control unit. The PIN code on the service card is covered and the user needs to scrape it in order to reveal it.

Authorized service partners can configure the control unit using the service code (on the service card). Features like motion detection and anti-hijack function can be disabled or enabled, and there are various parameters which can be fine-tuned by the authorized service provider.

2.4 Security claims by the manufacturer

According to the installation guide [9], Skybrake DD5 immobilizer employs "one of the most secure data coding algorithms AES-128". The user manual [8] further states that adversaries cannot disable the immobilizer system, even if he has captured data from the personal transceiver. It does not provide any further insights into the working principles, but such claims are plausible and there exist other car immobilizer systems using AES-128 like the Hitag AES which have not been broken and are considered secure [4].

The frequency range of the Skybrake DD5 (2400-2515 MHz[1]) might receive interferences from nearby cell phones or wireless networks. The Skybrake DD5 can switch between 126 radio channels in this frequency range and the manufacturer claims that the immobilizer will not be impacted by these interferences [8]. It seems that the manufacturer refers here to regular interferences and not jamming attacks targeted at the car immobilizer.

[1]Note that the installation guide [9] by the manufacturer erroneously defines the range as 2.4-2.4835 GHz

2.5 Installation of the car immobilizer

Figure 2 shows the diagram provided by the manufacturer for installation of the control unit in the car. The installation guide [9] recommends installing the control unit in a hidden place within the car's interior. It is recommended to place the buzzer far away from the control unit, as otherwise, it might reveal the location of the control unit.

The control unit is powered from the car's ignition and requires a working voltage between 7-30 VDC. The GND will be connected to the chassis of the car and the $+12\,\mathrm{V}$ power supply needs to be connected such that turning the ignition key will power the control unit. The control unit of the Skybrake DD5 has a relay switch with a normally closed (NC)[2] contact labelled as *blocking circuit*. It is common during the installation that either fuel pump, fuel valve or other common ignition circuits of the car are powered through the blocking circuit. This ensures that the engine will stop once the relay is powered and the contact is opened. Due to the blocking circuit using a normally closed relay switch, the car will remain drivable in case of power loss of the control unit, or in case of other malfunctions which prevent the control unit from powering the relay.

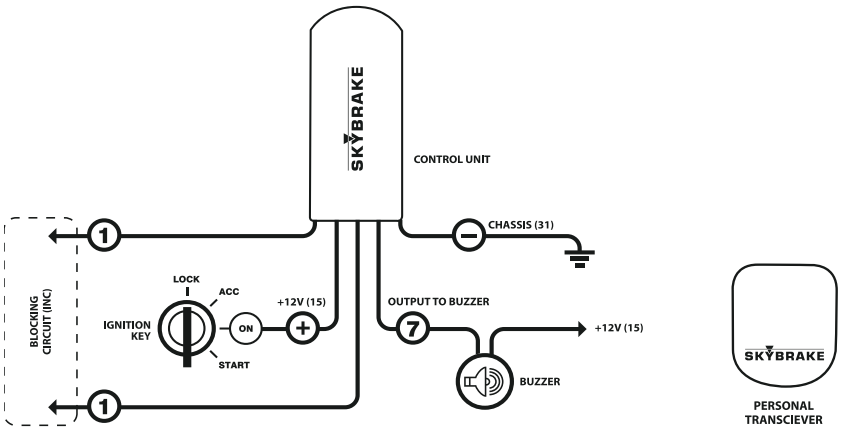


Figure 2: Connection diagram of immobilizer as shown in the installation guide [9]

Certified installers of the Skybrake DD5 immobilizer have special service equipment and can change the Skybrake DD5 settings of the control unit [9]. Various system time intervals can be configured and individual features like anti-hijack function or motion detection sensors can be disabled or enabled. The author of this work had access to one of the programming devices, but did not perform an in-depth analysis of the device and its interaction with the immobilizer. An overview of the programming device has been added to the Appendix I.

[2]Normally closed (NC): The contact is closed when no power is applied to the relay and opened if power is applied to the relay

3 Passive black-box analysis

This chapter covers the passive black-box analysis performed on the Skybrake DD5 car immobilizer. As part of the black-box analysis, the integrated circuit components of the immobilizer sets as well as the external behavior of the system during operation is observed. "Black-box" implies that the author had no knowledge about the system's internals, like, for example, firmware or memory contents of the microcontroller.

In this chapter, the car immobilizer is observed passively during its normal operation, without interfering with its operation by injecting or modifying any signals. The following Chapter 4 will cover additional black-box testing where signals are injected or modified in order to test various attack vectors.

3.1 Internal components of the car immobilizer

The aim of this section is to gain some high level understanding of the working principle by visually inspecting the exposed printed circuit board (PCB). The components and their datasheets can reveal information about the inner working of the car immobilizer. Additionally, signal traces, connection points or test pads, which can be leveraged later when doing a more thorough analysis, will be identified and highlighted.

3.1.1 Control Unit

Figure 3 shows the top of the exposed PCB board of the control unit. The heart of the control unit is a PIC24F32KA302 microcontroller by Microchip running at 32 MHz. The microcontroller has 32 KiB flash memory to hold program code, 512 bytes of EEPROM for configuration data and 2 KiB SRAM [10]. For radio communication with the personal transceiver, a nRF24L01+ transceiver chip by Nordic Semiconductor has been connected over an serial peripheral interface (SPI) bus.

Figure 4 shows the back of the PCB board of the control unit which exposes various test pads. The test pads expose the in-circuit serial programming (ICSP) interface of the microcontroller, as well as a UART debug interface.

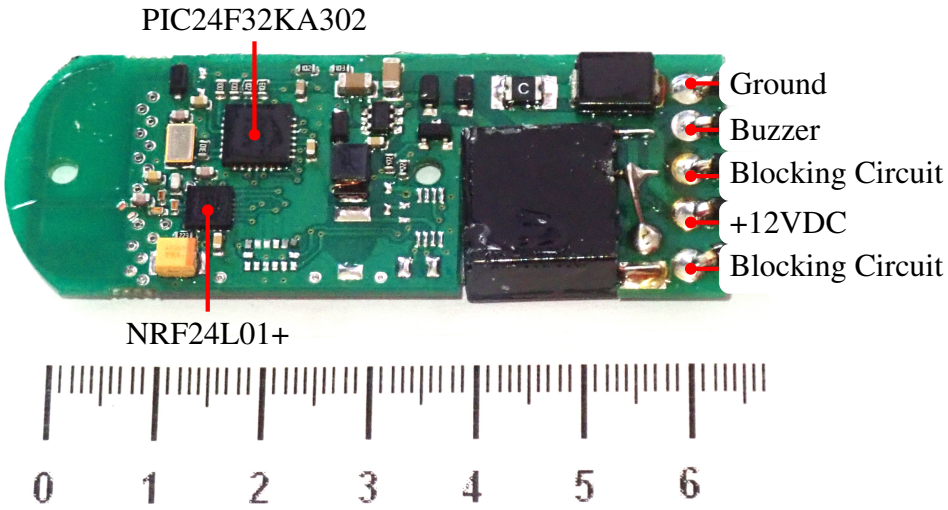


Figure 3: Front of exposed PCB of the control unit

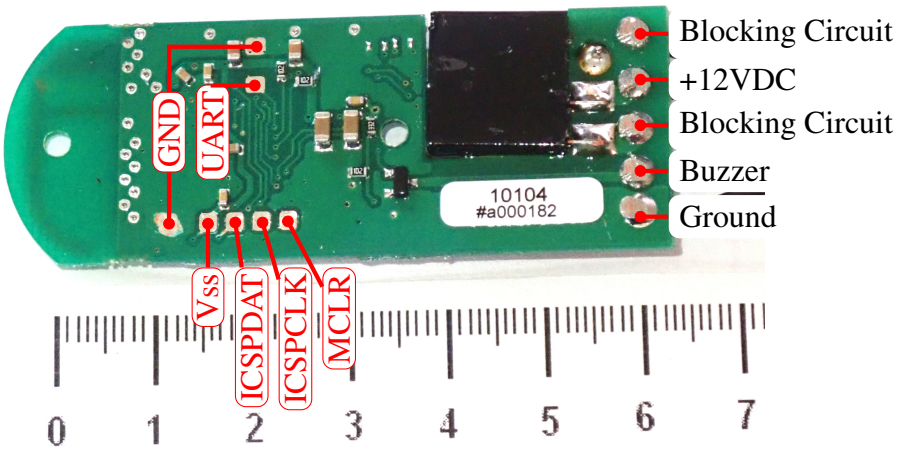
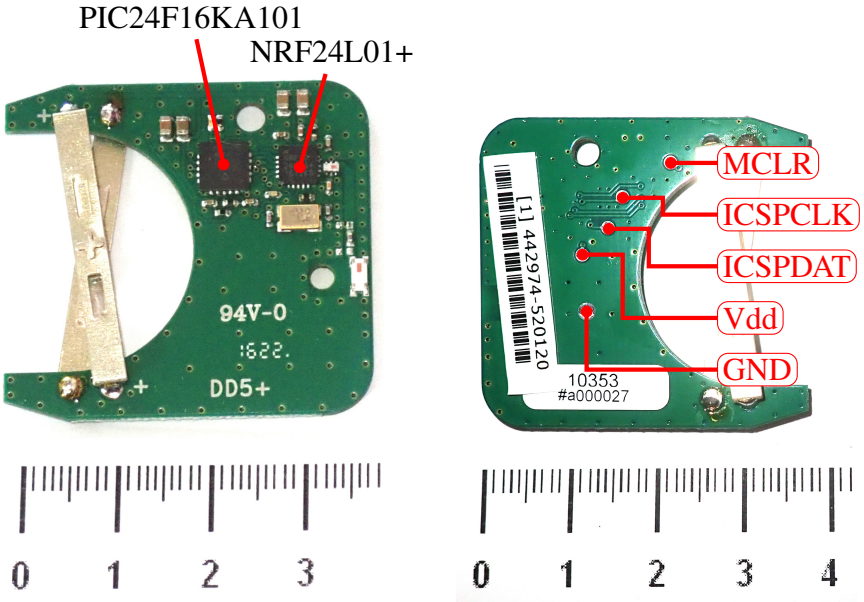


Figure 4: Back of exposed PCB of the control unit

3.1.2 Personal Transceiver

Figure 5a shows the personal transceiver of the car immobilizer. It is powered by a CR2430 battery. It is using an PIC24F16KA101 microcontroller by Microchip running at 32 MHz. The microcontroller has 16 KiB flash memory to hold program code, 512 bytes of EEPROM for configuration data and 1.5 KiB SRAM [11]. For radio communication with the control unit, it also uses a nRF24L01+ transceiver chip by Nordic Semiconductor that has been connected over an serial peripheral interface (SPI) bus.

The backside of the personal transceiver does not contain any components, but various test pads. The test pads expose the in-circuit serial programming (ICSP) interface of the microcontroller, see Figure 5b.



(a) Front of exposed PCB of the PT (b) Back of exposed PCB of the PT

Figure 5: Exposed PCB of the personal transceiver

3.2 UART debug interface

Surprisingly, a working serial port debug console was found on both control units that the author of this work had access to. It is unclear why the manufacturer preserved the serial interface meant for debugging purposes in the released product, as it might leak useful information to an adversary. Serial ports are configured with 115200 baud and use LF line termination instead of the more wide-spread CRCL termination. Using a USB to 3.3 V TTL adapter as shown in Figure 6, the data from the debug console can be captured using the `picocom` [12] command line tool with following arguments:

```
picocom /dev/ttyUSB0 -b 115200 --imap lfcrlf
```

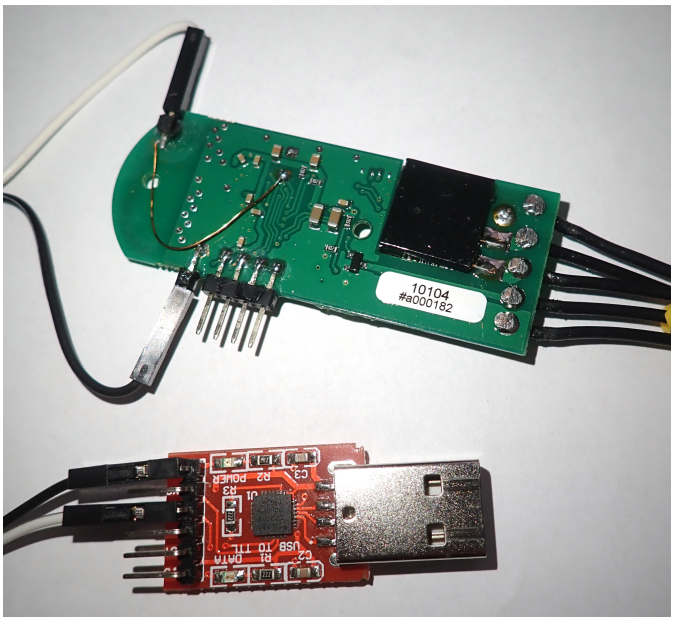


Figure 6: Setup to capture data UART debug console

The control unit logs various debug information during the startup. The actual output of the control units during startup has been attached in Appendix II. The control units run slightly different firmware versions and the output has slightly different formatting. Both immobilizer units log similar information and differences have been highlighted in Table 2. It is unclear whether the model 520120 and 520160 share the same firmware and the differences in formatting are solely due to the different firmware versions, or if there are different firmware variants for different models which are fundamentally different.

To the knowledge of the author of this work, there are no publicly available sources indicating how many Skybrake DD5 immobilizers were sold worldwide. The lowest Skybrake DD5 serial number seen by the author is $44080C_h$ [13], the highest $4457FC_h$. If we assume that the serial numbers are consecutive, it would give us a lower bound of at least 20 464 Skybrake DD5 immobilizers that have been produced.

Model	Serial number	SW version	Build date	HW revision
520120	442974_h	V22	2015-12-18 01:13:48	
520160	$4457FC_h$	V33	2018-01-26 15:04:25	7

Table 2: Firmware build information leaked over the UART debug console

3.3 Radio interface of car immobilizer

All components of the car immobilizer use the nRF24L01+ transceiver chip by Nordic Semiconductor to communicate with each other over radio. The nRF24L01+ transceiver chip transmits and receives messages over one of the 126 channels between 2400-2525 MHz using Gaussian Frequency Shift Keying (GFSK) modulation [14]. Each transmitted message can contain a user-defined payload of up to 32 bytes.

The nRF24L01+ transceiver chip makes use of logical addresses which are user-defined and can be changed through configuration registers [14]. Logical addresses are 3-5 bytes long, depending on the configuration. The transceiver chip will include the logical address with each message sent and will only receive messages addressed to the same logical address.

To ensure message integrity, the nRF24L01+ transceiver chip supports cyclic redundancy check (CRC) error-detecting codes. CRC checksums are 1-2 bytes long depending on the configuration. The transceiver chip will calculate and verify CRC checksums for all transmitted and received messages. Messages with invalid CRC will be automatically discarded by the transceiver chip.

Figure 7 shows the generic packet format which is transmitted by the transceiver chip. The Packet Control Field (PCF) contains fields that the nRF24L01+ transceiver chip uses for deduplication and automatic acknowledgement. The PID field is incremented with every message, and the nRF24L01+ transceiver chip automatically drops duplicate messages with identical CRC and PID. When the NO_ACK field is set, the nRF24L01+ transceiver chip will not send an automatic acknowledgement message once the package has been received.

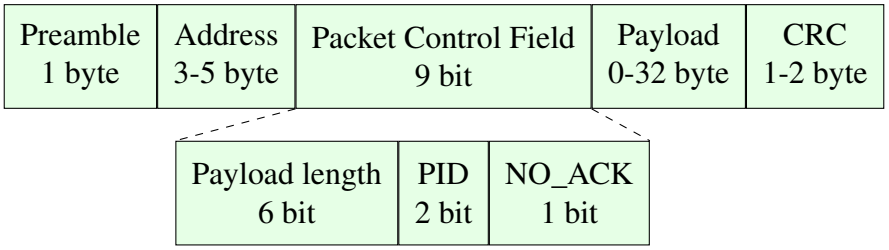


Figure 7: Generic packet format sent by transceiver chip nRF24L01+

The modulation scheme and packet structure can be reimplemented based on the datasheet [14] of the nRF24L01+ transceiver chip and a software defined radio (SDR) can be used to receive and send compatible messages. There exist already various open source decoders which implement decoders for the nRF24L01+ transceiver chip such as [15] or [16]. However, the car immobilizer may use any of the 126 available radio channels and might change the channel during its operation [8]. The changing of the

channel during operation will be referred to as *channel hopping* from here on. In order to capture all messages exchanged by the car immobilizer, a software defined radio with bandwidth of at least 126 MHz would be required in order to cover all radio channels simultaneously. This makes most commonly sold and available software defined radios, like for example the HackRF [17], unsuitable and therefore the author of this work did not pursue this idea any further.

3.3.1 Capturing messages from the SPI bus

The nRF24L01+ transceiver chip is a separate component on all the printed circuit boards which are connected to the microcontroller. The microcontroller communicates with the transceiver over serial peripheral interface (SPI) to configure it and to send or receive messages over the radio. Having physical access to the printed circuit board means it is possible to attach probes to the microcontroller pads or relevant PCB traces in order to decode the messages exchanged over the synchronous serial bus. Compared to capturing and decoding the exchanged messages from the radio signals, this simple approach has the benefit that it can capture all messages as seen by the microcontroller, without the need to know configuration of the nRF24L01+ transceiver chip like logical address or used radio channel frequency.

In order to analyze the communication protocol, a logic analyzer was directly attached to the SPI bus to the personal transceiver in order to capture MOSI, MISO, CLK and CS signals (see Figure 8a) between the PIC24F16KA101 microcontroller and the nRF24L01+ transceiver chip. The captured data was then analyzed using the command line tool `sigrok` [18] and its graphical user interface PulseView. The `sigrok` tool already has a built-in decoder for the protocol used by the nRF24L01+ transceiver chip (see Figure 8b), which can be used to convert the raw SPI communication to a more human-readable output. To capture data and decode the protocol for extended periods of time, the `sigrok` command line tool can be used with the arguments shown in Listing 1.

```
sigrok-cli --driver fx2lafw --continuous \
    --config samplerate=16m \
    -P spi:miso=D6:mosi=D4:clk=D2:cs=D0,nrf24l01
```

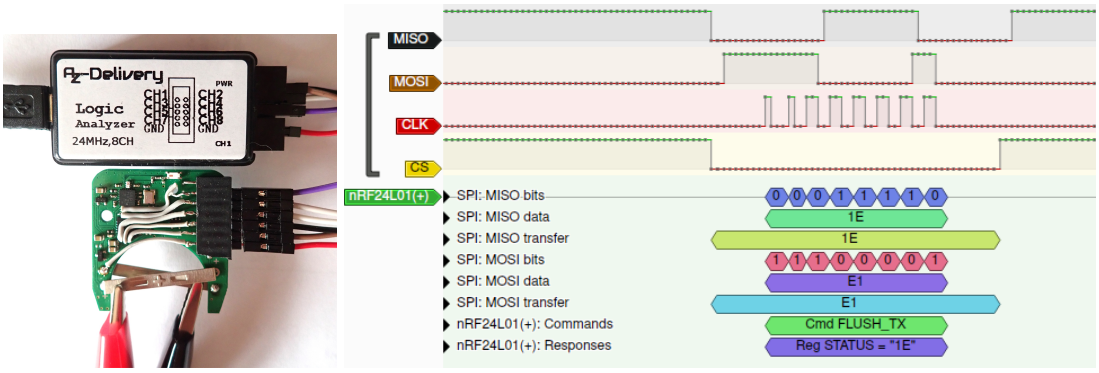
Listing 1: Command line arguments for sigrok to decode nRF24L01+ SPI command

Figure 9 shows the captured SPI traffic during one transmission event of the personal transceiver. Amongst other configuration related messages, it is possible to observe the received and sent message payloads of the transceiver chip. The write operations on various registers can be used to determine how the nRF24L01+ transceiver chip is configured. Based on the captured traffic and the datasheet [14] of nRF24L01+, the nRF24L01+ transceiver chip configuration of the personal transceiver has been fully reconstructed and is summarized in Table 3.

Address	Mnemonic	Bit	Value	Description
00_h	CONFIG			Configuration Register
	EN_CRC	3	1_b	Enable CRC
	CRC0	2	1_b	2 byte CRC encoding scheme
01_h	EN_AA			'Auto Acknowledgment' function
	ENAA_P0	0	1_b	Enable auto acknowledgement data pipe 0
02_h	EN_RXADDR			Enabled RX Addresses
	ERX_P0	0	1_b	Enable data pipe 0
03_h	SETUP_AW			Setup of Address Widths
	AW	1:0	10_b	4 bytes RX/TX address field width
04_h	SETUP_RETR			Setup of Automatic Retransmission
	ARD	7:4	1_h	Wait 500µS
	ARC	3:0	F_h	Up to 15 Re-Transmit on fail of AA
05_h	RF_CH			RF Channel
	RF_CH	6:0	04_h/$2F_h$/57_h[ab]	2404/2447/2487 MHz[ab]
			$1E_h$/49_h/73_h[ac]	2430/2474/2515 MHz[ac]
06_h	RF_SETUP			Setup of Address Widths
	RF_DR_HIGH	3	0_b[b] / 1_b[c]	1Mbps[b] / 2Mbps[c] high speed data rates
	RF_PWR	2:1	11_b	0dBm RF output power in TX mode
$0A_h$	RX_ADDR_P0	39:0	$1B61C5C5_h$[b]	Receive address data pipe 0
			$DC968FD7_h$[c]	
10_h	TX_ADDR	39:0	$1B61C5C5_h$[b]	Transmit address
			$DC968FD7_h$[c]	
$1C_h$	DYNPD			Enable dynamic payload length
	DPL_P0	0	1_b	Enable dynamic payload length data pipe 0
$1D_h$	FEATURE			Feature Register
	EN_DPL	2	1_b	Enables Dynamic Payload Length
	EN_ACK_PAY	1	1_b	Enables Payload with ACK
	EN_DYN_ACK	0	1_b	Enables W_TX_PAYLOAD_NOACK command

[a] Channels on which ping packets are sent for initiating handshake
[b] Used by immobilizer model 520120 with serial number 442974_h
[c] Used by immobilizer model 520160 with serial number $4457FC_h$

Table 3: Configuration registers of nRF24L01+ transceiver of PT



(a) Logic analyzer (b) Pulseview application

Figure 8: Setup for capturing data from the SPI bus

The main features which have been enabled on the nRF24L01+ transceiver chip will be shortly highlighted below:

Cyclic Redundancy Check (CRC) is enabled to ensure integrity of the message. A 2-byte CRC is appended to each message. The polynomial $x^{16} + x^{12} + x^5 + 1$ with initial value $\texttt{FFFF}_\text{h}$ is used. The CRC is calculated over the address, packet control field and the payload. The nRF24L01+ transceiver chip will automatically verify the CRC and discard messages with an invalid CRC.

Dynamic Payload Length (DPL) is enabled, which means the package control field will include a 6-bit payload length field specifying the length in bytes of the message payload. However, all observed messages that have been sent by the immobilizer always contained 16 bytes of payload, so it is likely that all the exchanged messages have a fixed length of 16 bytes.

Acknowledgment with payload is enabled, which means that an acknowledgment message sent in response to a received message can include a user-defined payload. However, due to the enabled `EN_DYN_ACK` mnemonic, the transceiver chip will also allow transmitting of messages which do not have automatic acknowledgement by setting a special flag `NO_ACK` in the transmitted message. All observed messages sent by the immobilizer did have the bit flag `NO_ACK` set and no automatic acknowledgement messages were observed.

4-bytes address fields are being used. The two immobilizer models available to the author of this work use different addresses, meaning that each immobilizer set uses a unique logical address for its communication between the personal transceiver and control unit. Unfortunately, there seems to be no obvious correlation between the serial number, service code and address. However, the programming device (see Appendix I) needs to determine the logical address based on serial number

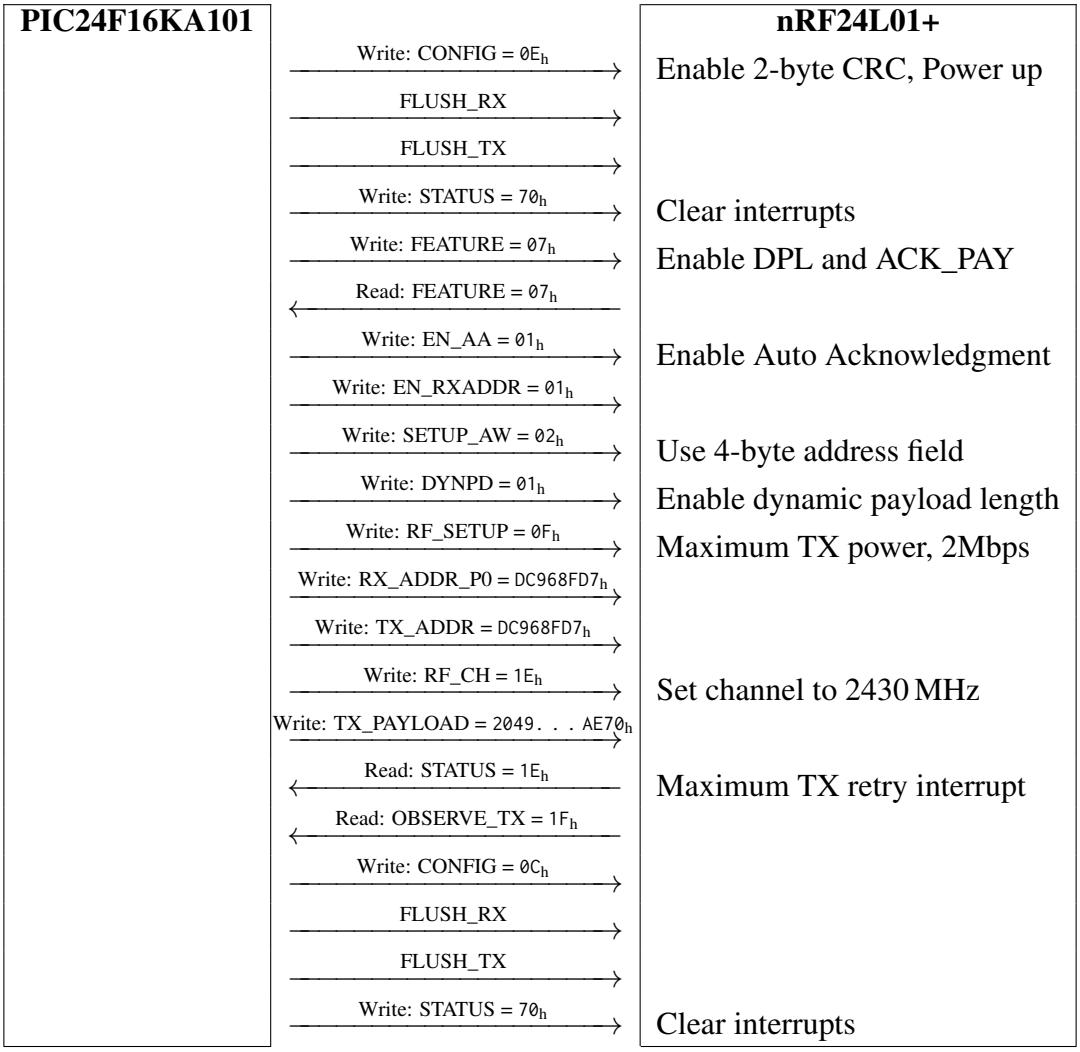


Figure 9: Example of captured SPI traffic on personal transceiver

and service code, as otherwise it would not know what address to use for communication with the control unit. Therefore it is likely that the address is derived somehow from the serial number, service code or both.

Having observed all configuration registers of the nRF24L01+ transceiver chip, the exact packet format transmitted by the transceiver chip over the air can be reconstructed. Figure 10 shows the packet format of packets exchanged by the car immobilizer.

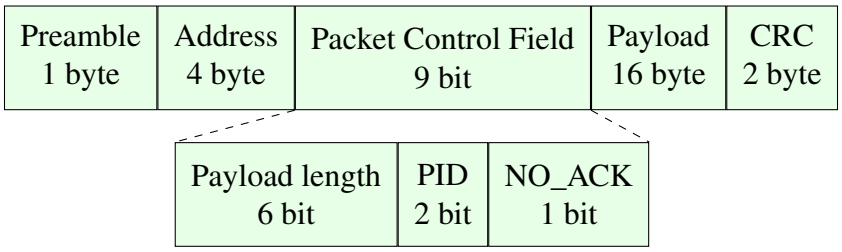


Figure 10: Packet format sent by Skybrake DD5 immobilizer

3.3.2 Capture and transmit messages using nRF52840 USB Dongle

In order to capture and send messages from and to the nRF24L01+ transceiver chip, the nRF52840 USB dongle from Nordic Semiconductor (see Figure 11) can be used. Unlike attaching a logic analyzer to the signal traces of the PCB, this approach does not require physical access, is less intrusive and therefore does not pose any risk of damaging any of the components of the immobilizer. Being able to send messages to the car immobilizer will be useful for testing possible attacks, which will be performed in Chapter 4.

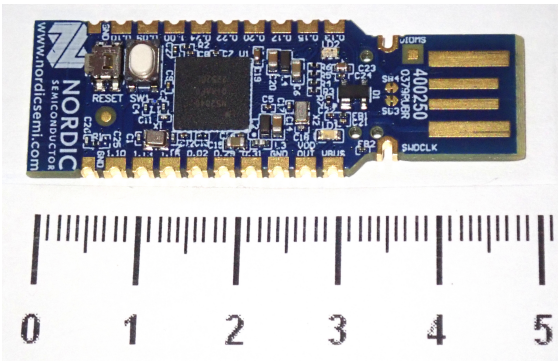


Figure 11: nRF52840 USB Dongle by Nordic Semiconductor

The nRF52840 system on chip (SOC) consists amongst other functionality of an ARM Cortex microprocessor running at 64 MHz with 1 MiB flash and 32 KiB RAM and a 2.4 GHz radio transceiver which is able to transmit and receive packages from the

nRF24L01+ transceiver chip when configured correctly [19]. The general packet format which is supported by the nRF52840 is shown in Figure 12.

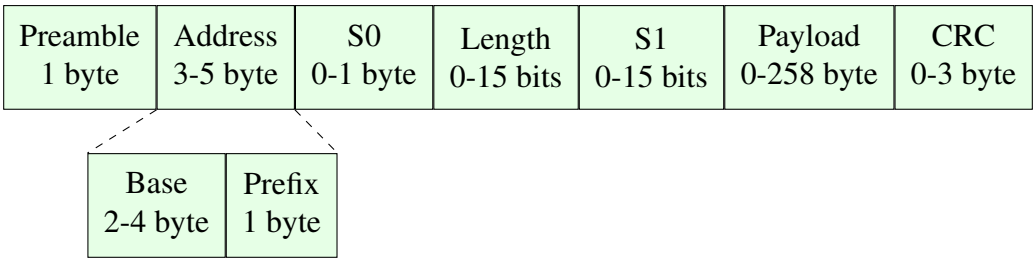


Figure 12: Generic packet format sent by nRF52840 USB Dongle

Configuring the registers as shown in Table 4 will result in the packet format shown in Figure 13. This is fully compatible with the nRF24L01+ transceiver chip configuration used by the car immobilizer. Note that the nRF52840 chip does not have hardware support for many features offered by the nRF24L01+ transceiver chip, like support for automatic acknowledgment, retries, deduplication, and discarding of messages with invalid CRC. Those features would need to be implemented in software if needed. That said, the nRF52840 USB dongle is more than sufficient for the purpose of interacting with the Skybrake DD5 car immobilizer and to perform simple capturing and transmissions during experiments.

Note that one major drawback is that the nRF52840 USB dongle needs to be configured with the correct radio channel and logical address matching the configuration of the car immobilizer. This means that it is not possible to capture traffic over the radio when channel hopping is used and the channel selection algorithm is unknown.

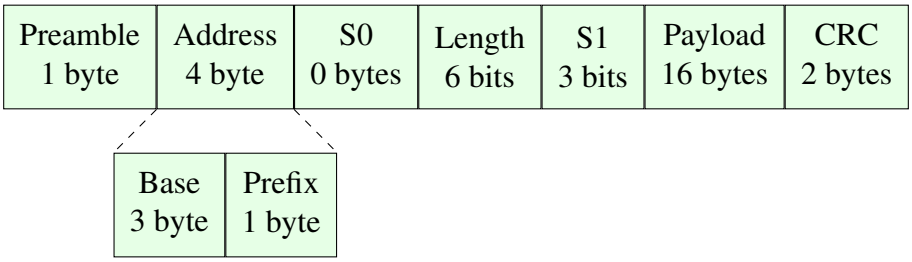


Figure 13: nRF52840 packet format compatible with Skybrake DD5 immobilizer

Address	Mnemonic	Bit	Value	Description
508_h	FREQUENCY			
	FREQUENCY	6:0	$04_h/2F_h/57_h$[ab]	2404/2447/2487 MHz[ab]
			$1E_h/49_h/73_h$[ac]	2430/2474/2515 MHz[ac]
$50C_h$	TXPOWER		00_h	0 dBm
510_h	MODE		0_h[b]/1_h[c]	$1^b/2^c$ Mbps Nordic proprietary radio mode
514_h	PCNF0			Packet configuration register 0
	S1LEN	19:16	3_h	3 bits long S1 field
	S0LEN	8	0_b	0 bytes long S0 field
	LFLEN	3:0	6_h	6 bits long LENGTH field
518_h	PCNF1			Packet configuration register 1
	ENDIAN	24	1_b	Big Endian: Most significant bit on air first
	BALEN	18:16	3_h	3 bytes of base address length
	MAXLEN	7:0	20_h	32 bytes of maximum packet payload
$51C_h$	BASE0		$C5611B00_h$[b]	Base address 0
			$8F96DC00_h$[c]	
524_h	PREFIX0			Prefixes bytes for logical addresses
	AP0	7:0	$C5_h$[b] / $D7_h$[c]	Address prefix 0
530_h	RXADDRESSES			
	ADDR0	0	1_b	Enable reception on logical address 0
534_h	CRCCNF			CRC configuration
	LEN	1:0	2_h	Enable CRC calculation with length of 2 bytes
538_h	CRCPOLY		11021_h	CRC polynomial: $x^{16} + x^{12} + x^5 + 1$
$53C_h$	CRCINIT		$FFFF_h$	CRC initial value
650_h	MODECNF0			
	RU	0	1_b	Enable fast radio ramp-up

[a] Channels on which ping packets are sent for initiating handshake
[b] Used by immobilizer model 520120 with serial number 442974_h
[c] Used by immobilizer model 520160 with serial number $4457FC_h$

Table 4: Configuration registers of nRF52840 chip compatible with immobilizer

3.4 Capturing exchanged message

To observe what messages are sent by the personal transceiver, a logic analyzer connected to the SPI bus on the personal transceiver as shown in Section 3.3.1 was used. This is the most reliable method to capture all exchanged messages as the personal transceiver takes care of the correct configuration, in particular of the correct radio channel.

3.4.1 Ping message

When no control unit in the vicinity is responding to the personal transceiver, the default behaviour of the personal transceiver is to periodically send messages. Messages that are sent by the personal transceiver while being unauthenticated will be referred to as *ping messages* from now on[3]. Figure 14 shows the behavior and captured traffic of an unauthenticated personal transceiver.

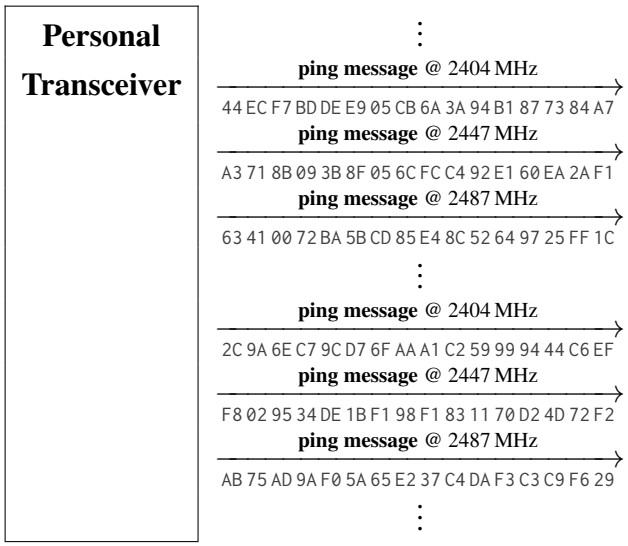


Figure 14: Encrypted messages from unauthenticated personal transceiver

The unauthenticated personal transceiver will send ping messages round-robin on three different radio channels. Each message has a random looking payload of 16 bytes. After each sent ping message, personal transceiver will wait for up to 2 ms for a response on the same radio channel. The messages are sent in bursts to all three radio channels every 5 seconds. The payload of the messages sent to different channels are different. It is likely that the radio and the microcontroller are put to a standby state in between the bursts in order to conserve energy. Sending the ping message on different channels most

[3]This matches the nomenclatures used by the manufacturer, as the UART debug serial console also refers to those messages as *PT ping packets*.

likely is done to limit the impact of interferences from other radio devices like mobile phones or wireless networks.

The used channels for sending the ping messages differ amongst the two car immobilizer sets that were analyzed. The immobilizer model 520120 with serial number 442974_h will send ping messages to channels 04_h (2404 Mhz), $2F_h$ (2447 Mhz) and 57_h (2487 Mhz), whereas the immobilizer model 520160 with serial number $4457FC_h$ will send ping messages to channels $1E_h$ (2430 Mhz), 49_h (2473 Mhz) and 73_h (2515 Mhz). The term *handshake channel* is used to refer to the three channels receiving the initial ping messages.

The service card which is contained within the immobilizer cardboard box contains a 128-bit long hexadecimal encoded service code. It turns out that this is indeed the encryption key for the messages sent by the personal transceiver. The exchanged messages are using AES-128 encryption in ECB mode and they can be decrypted by using the service code (from the service card) as the AES 128-bit key. Figure 15 shows the messages after decryption, which with the exception of the last 3 bytes are all identical.

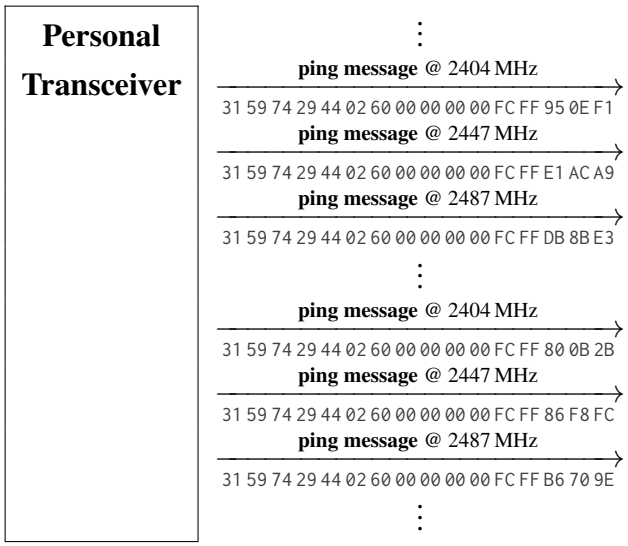


Figure 15: Decrypted messages from unauthenticated personal transceiver

3.4.2 Authentication

When the car ignition is turned on, the UART debug output of the control unit show that the control unit will start listening for incoming messages. The control unit will listen for incoming messages for the duration of 3 seconds before changing round-robin to the next channel, using the same three channels where the personal transceiver transmits the ping messages.

When the control unit receives a ping message from the personal transceiver, it will respond on the same channel with a message that has a random looking payload of 16 bytes. Messages sent from the control unit to the personal transceiver will be referred to as *challenge request* from now on. After a challenge request has been sent by the control unit, it will change to a seemingly random new radio channel and listen for the duration of 3 seconds for incoming messages.

Once the personal transceiver receives a challenge request, it will change to the same radio channel as the control unit and transmit a message that has a random looking payload of 16 bytes. Messages which are sent in response to challenge requests will be referred to as *challenge response* from now on. The personal transceiver will immediately respond to the first challenge request received on one of the three handshake channels. The control unit will keep responding to the personal transceiver with new challenge requests which the personal transceiver has to answer on a new seemingly random channel. The personal transceiver will send responses to following challenge requests with a delay of 3 seconds. This is most likely to reduce power consumption on the personal transceiver and allow for time to put both radio and microcontroller into standby mode.

All the random looking messages are also encrypted with AES-128 using the service code of the immobilizer as an encryption key. Figure 16 shows the observed message flow after decryption, which again reveals noticeable patterns that allow reconstruction of the plaintext message structure.

Upon receiving the first challenge response message, the control unit moves to the authenticated state. This means that it will not immobilize the car, or it will undo the immobilization if it was previously immobilized. The control unit will answer each challenge response with a new challenge request, which the personal transceiver needs to answer on a new seemingly random channel. The continuous challenge-response message exchange allows the control unit to monitor for the presence of the personal transceiver, which is a requirement for the Anti-Hijack feature.

If the message exchange during the authenticated state is disturbed by interferences, i.e. any challenge request or challenge response is lost, the whole authentication process is starting from the beginning and the personal transceiver will start over by sending ping messages on the handshake channels.

During the authentication, the UART debug console (see Appendix III) reveals additional information contained in the payload of the challenge response. With each challenge response, the personal transceiver sends the coin battery voltage (with millivolt resolution). Additionally, the control unit keeps track of a sequence number within its memory which is initialized with 1 and incremented with each challenge response.

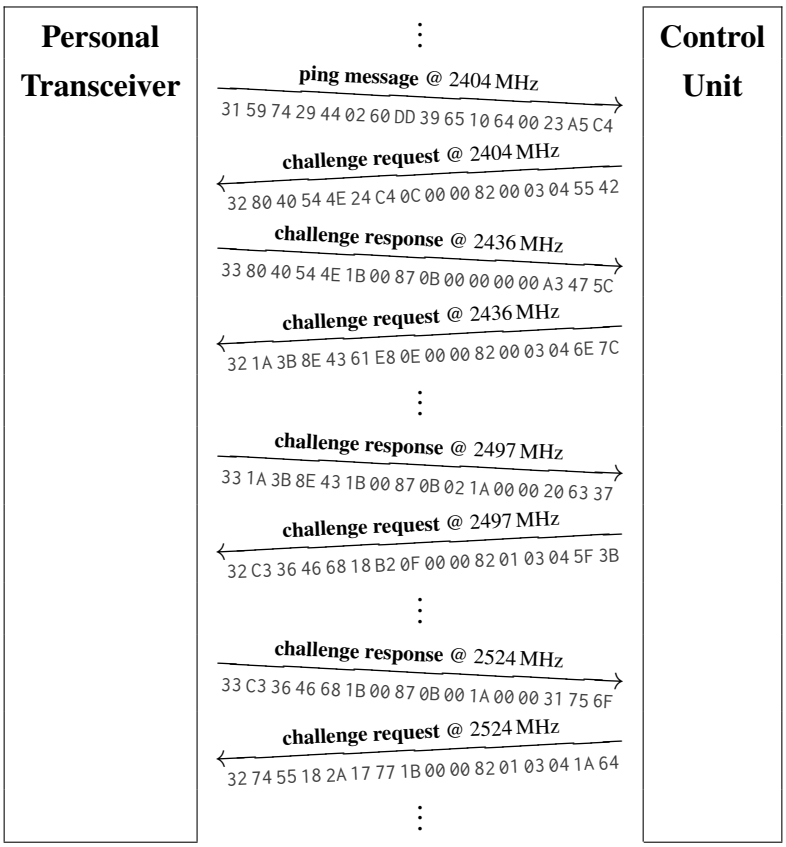


Figure 16: Captured and decrypted authentication messages between CU and PT

3.5 Statistical analysis

The statistical analysis in this section was performed by the author of this work prior discovering that the messages can be decrypted using the service code as AES-128 key. Therefore, all analysis was performed on ciphertexts rather than plaintexts.

The exchanged messages between personal transceiver and control unit are AES-128 encrypted and have clear observable structures after decryption. The ciphertexts of the exchanged messages look random, because all plaintext messages include some bytes which are randomly chosen.

This section explores bigger data sets of captured ciphertext messages between the control unit and the personal transceiver in order to see how randomness is selected and how channels for exchanged messages are selected.

3.5.1 Duplicate ping messages sent by personal transceiver

Over the period of approximately 24 hours, 157 967 encrypted ping messages were gathered, of which 156 876 were unique. That means there were 1 091 duplicate encrypted ping messages sent by the personal transceiver.

If the plaintext data included a sequence number, duplicate messages would occur once the sequence number overflows. For a n-bit sequence number, only the first 2^n messages would be unique, followed by duplicates. From previously captured and decoded messages, we know this assumption is wrong. The data (see Figure 17) clearly shows a non-linear increase in duplicate messages over time, which is not consistent with the assumption that the plaintext contains a sequence number.

Analogous to the birthday paradox problem, using the assumption that the personal transceiver has n bits in the plaintext which are randomly chosen, the number of expected unique n-bit sequences in the data set can be calculated as shown in equation (1).

$$c_{\text{uniq}} = 2^n - 2^n \cdot \left(\frac{2^n - 1}{2^n}\right)^m = 2^n \cdot \left(1 - \left(\frac{2^n - 1}{2^n}\right)^m\right) \tag{1}$$

Having the number of observed messages and the number of expected unique messages, the number of sent duplicates c_{dup} can be calculated as shown in equation (2).

$$c_{\text{dup}} = m - c_{\text{uniq}} = m - 2^n \cdot \left(1 - \left(\frac{2^n - 1}{2^n}\right)^m\right) \tag{2}$$

Based on the comparison of observed versus the expected duplicates (see Figure 17), it indeed looks likely that the personal transceiver includes some randomness in the initial

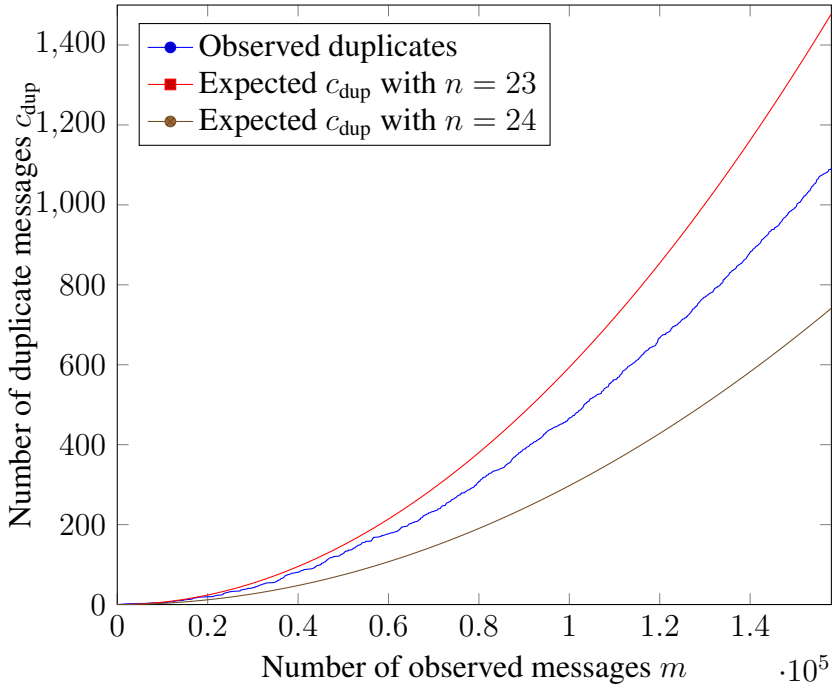


Figure 17: Duplicate ping messages sent by personal transceiver

ping message. The initial ping message sent by the personal transceiver must have at least 24 bits in the plaintext which are variable, any lower number of bits would have resulted in more duplicate messages. This is indeed what the previously captured ping message confirms, where the last three bytes are randomly chosen.

Both tags belonging to the same car immobilizer set will transmit duplicates, but it was never observed that any single message was sent by both personal transceivers. This means that the set of ping messages sent by the two personal transceivers are dis-joint and there are no common messages amongst them. This makes it very likely that the plaintext contains an identifier distinguishing the two personal transceivers. This is indeed what can be observed in the captured plaintext, where the 6th byte will either contain $\texttt{01}_\text{h}$ or $\texttt{02}_\text{h}$, depending on the which personal transceiver has sent the ping message.

No duplicates have been observed for challenge requests and responses, indicating that the entropy in the plaintext message for those messages is significantly higher. And indeed, in addition to the last three random bytes of each message, the challenge request and response contain an additional 4 bytes (2nd-5th byte) which are part of the challenge and are randomly chosen.

3.5.2 Pseudorandom number sequence test

The observed messages should look completely random, because AES-128 acts as a pseudorandom permutation. To confirm this hypothesis, the pseudorandom number sequence test program `ent` [20] has been used.

	Entropy [bits/byte]	Chi-square	Mean	Monte-Carlo-Pi	Serial-Correlation
Ping[a]	7.999922	272.17	127.4915	3.142788638	-0.000935
Challenge Request[b]	7.999749	261.96	127.5894	3.126241413	0.000536
Challenge Response[c]	7.999838	218.98	127.5955	3.142065749	-0.001603

[a] Sample of 157967 ping messages sent by personal transceiver were used
[b] Sample of 61246 challenge requests sent by control unit were used
[c] Sample of 60788 challenge responses sent by personal transceiver were used

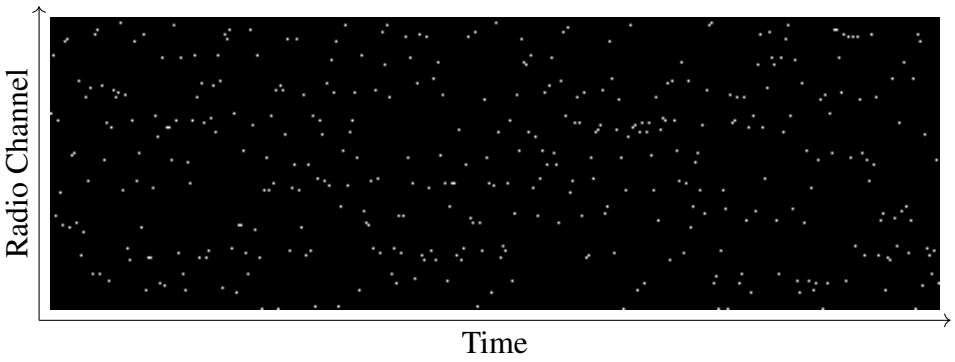
Table 5: Output of pseudorandom number sequence test program `ent`

The messages exchanged between the personal transceiver and the control unit are seemingly random as shown in Table 5. The probability that a random sequence would exceed the Chi-square value was between 5% and 95% for all message sets, which means the sequences are not suspect and behave like a random source of data [21].

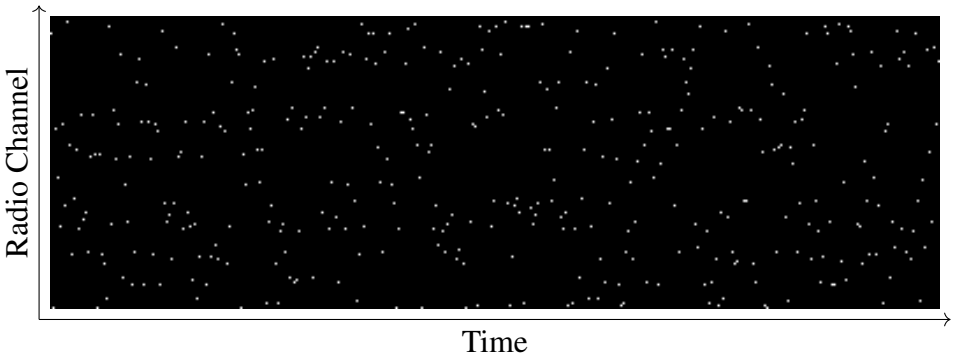
3.5.3 Channel Hopping

In absence of a control unit, the personal transceiver will round-robin send ping messages on three different channels. However, when the control unit is in the vicinity, the challenge-response messages are exchanged on *seemingly* random radio channels. But, the radio channel cannot be random, the personal transceiver and control unit need to agree upon one single radio channel in order to successfully exchange messages. Indeed it can be observed that the 6th byte of the challenge request is the channel where the response is expected. This section explores patterns of selected radio channels in order to gain additional insights into how the radio channel is selected by the control unit.

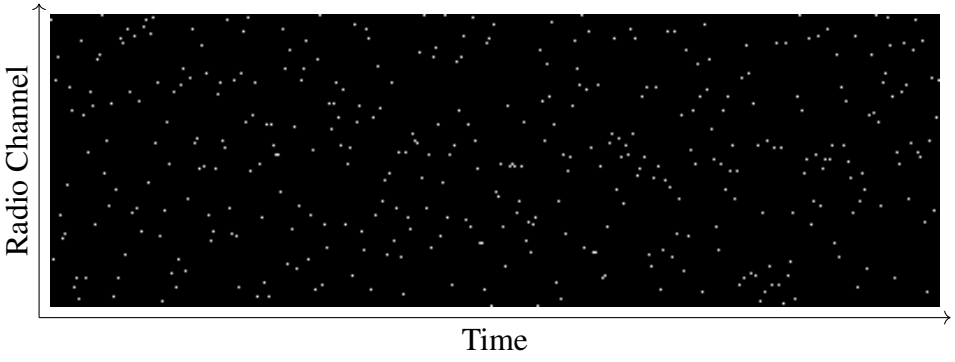
Figure 18a and Figure 18b shows a dot plot chart of the selected radio channels for exchanged messages between the personal transceiver and the control unit. For each exchanged message, one column is added to the dot plot chart highlighting the selected radio channel. For comparison, Figure 18c has been added where all the channels are uniformly randomly chosen between the 126 available radio channels. `/dev/urandom` was used as a source of randomness with rejection sampling to ensure all channels are in the valid range $\texttt{00}_\text{h}$-$\texttt{7D}_\text{h}$ of the nRF24L01+ transceiver chip.



(a) Car Immobilizer model 520120 with serial number $\texttt{442974}_\text{h}$



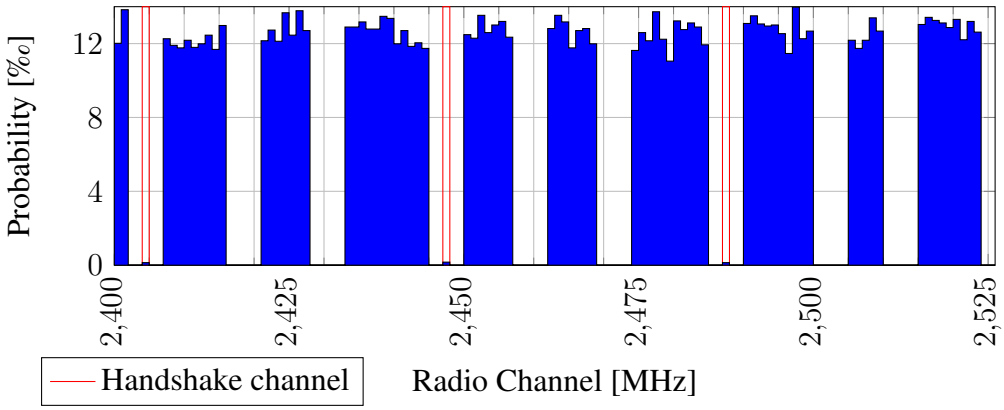
(b) Car Immobilizer model 520160 with serial number $\texttt{4457FC}_\text{h}$



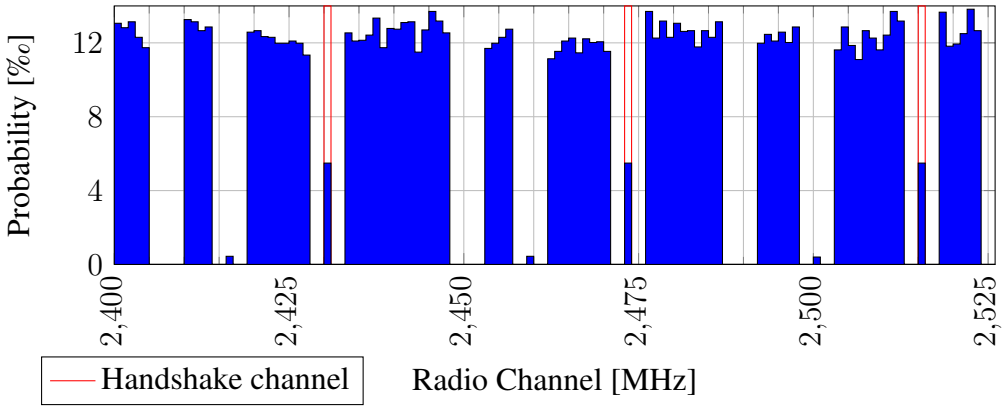
(c) Expected dot plot chart if channel selection is uniformly random

Figure 18: Dot plot chart of selected radio channels

Looking at the dot plot charts in Figure 18, an extremely observant reader might notice that the distribution of the selected channel by the car immobilizer does not look uniformly random. Indeed, certain channels are never chosen which becomes obvious when looking at a histogram of selected channels as shown in Figure 19.



(a) Immobilizer model 520120 with serial number $\texttt{442974}_\text{h}$



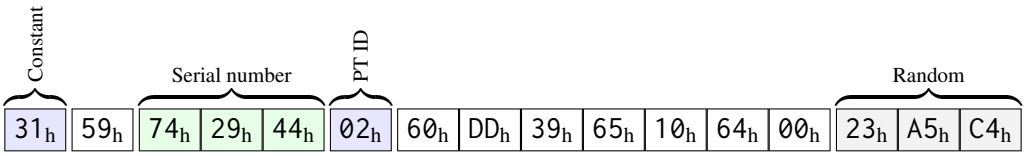
(b) Immobilizer model 520160 with serial number $\texttt{4457FC}_\text{h}$

Figure 19: Histogram of selected radio channels

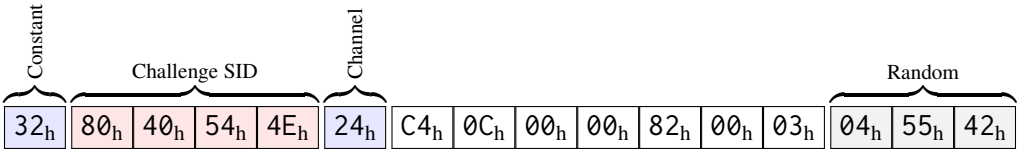
The immobilizer model 520120 with serial number $\texttt{442974}_\text{h}$ only uses 79 distinct channels out of 126 available channels for exchanging messages, while the immobilizer model 520160 with serial number $\texttt{4457FC}_\text{h}$ uses 82 distinct channels. The three channels on which the personal transceiver sends the ping messages have noticeably lower traffic. Based on the captured data, the control unit will never expect any challenge responses on these channels and they are only used when interferences forced re-authentication between the control unit and personal transceiver, which would restart the whole authentication process starting with ping messages.

3.6 Conclusions of passive black-box analysis

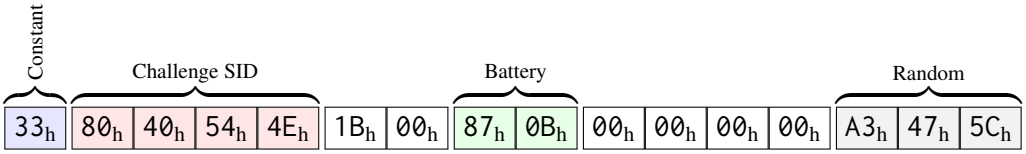
It was discovered that the random looking 16 bytes long messages exchanged by the immobilizer are indeed AES-128 encrypted. The service code from the service card is used as an encryption key, which means that each car immobilizer has their own supposedly unique key. Knowing the service code allows for decryption of the exchanged messages and reveals clear patterns within the individual message types. Figure 20 describes the discovered fields in the different message types. The message type itself is indicated by the first byte. All messages appear to have the last three bytes randomized. Note that the serial number and battery voltage is encoded using big endian, having the least significant byte first and the most significant byte last.



(a) Ping message sent from personal transceiver to control unit



(b) Challenge request sent from control unit to personal transceiver



(c) Challenge response sent from personal transceiver to control unit

Figure 20: Plaintext message format of exchanged messages

In order to authenticate, the personal transceiver needs to send the correct message type and repeat the random SID received in the challenge request. The personal transceiver therefore needs to be able to decrypt the challenge request and encrypt the challenge response using the service code. This functionality of the personal transceiver during normal operation can be fully emulated on a nRF52840 USB dongle and it is possible to pass authentication with the control unit without the real personal transceiver if the

service code, serial number and logical address of the immobilizer is known. If only the service code is known, then the logical address can be captured by using a software defined radio[4], and the serial number can be obtained by decrypting the ping message if the personal transceiver is in the vicinity.

[4]Note that the logical address cannot be captured using the nRF52840 or nRF24L01+ transceiver chip, as they would need to be configured with the correct logical address in the first place in order to capture relevant messages.

4 Overview of possible attack vectors

This section covers various attack vectors for car immobilizers and evaluates if the Skybrake DD5 immobilizer might be vulnerable to them or not. Various experiments where messages will be injected or modified will be performed to establish whether the car immobilizer is vulnerable to described attacks. This testing gives additional insights into the working principle of the Skybrake DD5 immobilizer.

Note that any message exchange diagrams in this section will contain the message in encrypted form, as it is expected that an adversary does not have access to the service code on the service card and therefore will not be able to decrypt the messages.

4.1 Leakage of service code

As discovered in the previous chapter, the immobilizer exchanges AES-128 encrypted messages which use the service code printed on the service card as an encryption key. This service code is the only secret needed by an adversary in order to be able to replicate the authentication functionality of the personal transceiver and successfully authenticate with the control unit. The design of the service card (see Figure 21) conveys to the user that the PIN code, which needs to be scraped to reveal, needs to be kept secret at all times. Given that the service code allows an adversary to authenticate with the immobilizer, the service code also must be kept secret at all times. Unfortunately, the service card does not provide any indication to the user that the service code must be kept secret. Searching the internet for Skybrake DD5 immobilizer that are being sold reveals results like [13], [22], [23] or [24] where sold immobilizers are depicted with their service card containing the service code. Purchasing immobilizers online from untrusted sources exposes the buyer to some risk. The seller is able to learn the service code, since the packaging of the car immobilizer does not have any security seal and, unlike the PIN code, the service code is not protected by any means on the service card. Additionally, through shipping information, the seller might learn about the residential address where the buyer's car is most likely located, making a car theft feasible.

(a) Model 520120 with serial number 442974_{h} (b) Model 520160 with serial number 4457FC_{h}

Figure 21: Service cards of the Skybrake DD5 immobilizers used in this work

4.2 Replaying attacks

Replaying attacks are attacks where an adversary intercepts and stores valid data transmission in order to re-transmit them at a later time. In the context of a car immobilizer, there must be countermeasures which prevent an adversary from replaying previously captured challenge responses from the personal transceivers in order to pass the authentication.

4.2.1 Replaying ping message

Replaying ping messages of the same immobilizer set would allow an adversary to be able to initiate the authentication process and capture the first challenge requests from the control unit.

The outlined experiment in this section will first capture a genuine authentication flow between the personal transceiver and the control unit using a logic analyzer connected to the SPI bus on the personal transceiver as shown in Section 3.3.1. Afterwards, the personal transceiver will be disabled and an identical copy of the recorded ping message using a nRF52840 USB Dongle will be sent to the control unit.

Figure 22 shows the message flow of the performed experiment, including the captured messages themselves. This attack can be performed successfully, the control unit accepts the replayed ping message as would be the case if it was genuine. It can be observed that, even for identical ping messages, the control unit will send a different challenge to the personal transceiver. The UART debug information shows that the control unit listens on a different channel for the challenge response by the personal transceiver. This is what is expected, as it was discovered that the control unit is in charge of selecting the channel for the challenge response in the previous chapter, and it does this randomly independent of the received ping message.

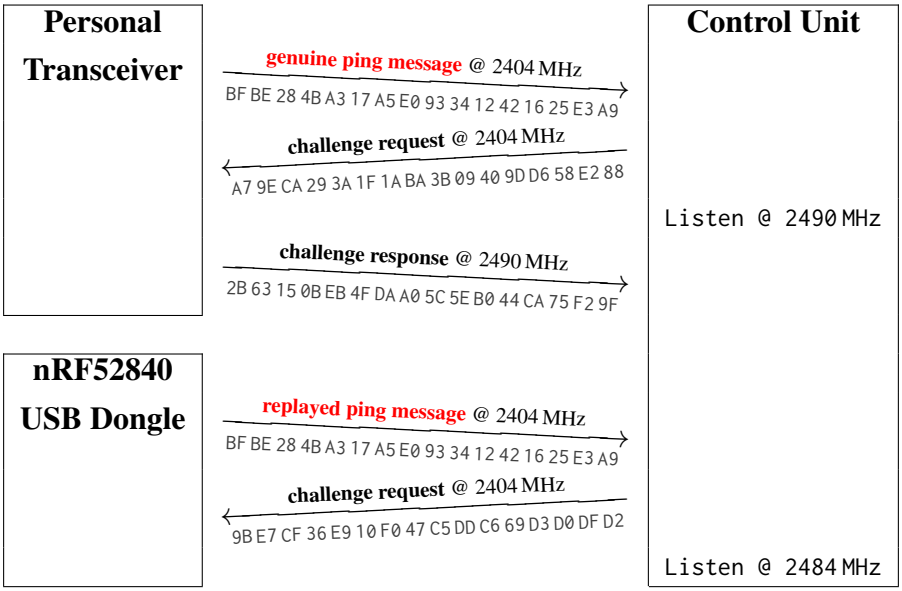


Figure 22: Replay attack for ping messages

Note that replaying only the ping message will not allow the adversary to bypass the immobilizer. The control unit will only unblock the blocking circuit upon receiving the first valid challenge response. Extending this replay attack by simply replaying previously captured challenge responses is not possible. An adversary cannot simply replay previously captured messages on the same radio channels. Without access to the service code to decrypt the challenge request, an adversary would not know the correct channels where to send the challenge response.

4.2.2 Replay challenge responses

As previously shown, simply replaying challenge responses of the personal transceiver is not possible, as the control unit will select seemingly random channels on which the challenge-response messages have to be exchanged on.

However, the control unit will wait for up to 3 seconds for each challenge response message. This is a very long time and an adversary might simply "brute-force" the channel, i.e. broadcast the challenge response to all 126 available radio channels.

The time it takes to broadcast a message to all 126 radio channels can be estimated using the datasheets [14] and [19]. For the immobilizer with the serial number 442974_{h} using 1 Mbps data rate, the on-air time t_{oa} for a single transmitted message can be calculated as shown in equation (3).

$$t_{oa} = 8\,{}^{\text{bits}}/_{\text{byte}} \cdot \frac{(\overbrace{1\,\text{byte}}^{\text{preamble}} + \overbrace{4\,\text{bytes}}^{\text{address}} + \overbrace{16\,\text{bytes}}^{\text{payload}} + \overbrace{2\,\text{bytes}}^{\text{CRC}})}{1\,000\,000\,{}^{\text{bits}}/_{\text{s}}} + \overbrace{9\,\text{bits}}^{\text{PCF}} = 193\,\mu\text{s} \quad (3)$$

If the nRF24L01+ transceiver is used by the adversary, additional delay t_{ul} is introduced by reuploading of the payload to the nRF24L01+ transceiver chip over the SPI bus. Assuming the highest bus speed of 10 Mbps specified in the datasheet [14] is used, the additional delay can be calculated as shown in equation (4). When using the nRF52840 USB dongle, no such additional delay is introduced.

$$t_{ul} = \frac{8\,{}^{\text{bits}}/_{\text{byte}} \cdot \overbrace{16\,\text{bytes}}^{\text{payload length}}}{10\,000\,000\,{}^{\text{bits}}/_{\text{s}}} = 12.8\,\mu\text{s} \quad (4)$$

Additionally, for each packet there is additional time required to disable and re-enable the radio on the changed radio frequency as outlined in the datasheets [14] and [19]. The approximate resulting total time has been summarized in Table 6.

	Upload	**Enable**	**Transmit**	**Disable**	**Total**
nRF24L01+	12.8 µs	130 µs	193 µs		335.8 µs
nRF52840		40[a] / 140[b] µs	0 µs	6 µs	46[a] / 146[b] µs

[a] With fast-ramp up enabled
[b] With fast-ramp up disabled

Table 6: Required time to transmit single message

Broadcasting a single message to all available 126 radio channels using the slower nRF24L01+ transceiver chip thus takes at least $126 \cdot 335.8\,\mu\text{s} = 42.31\,\text{ms}$. This is order of magnitudes smaller than the available time of 3 seconds and an adversary does not need to know the correct channel for the response and can simply broadcast the message to all available 126 channels.

This means replay attacks can be performed where the challenge response is sent to all available 126 radio channels after the challenge request has been received. Figure 23 shows one of the performed attacks by the author. While the control unit will successfully receive the replayed challenge response, it will not transition into the authenticated state. The UART debug interface reveals that the control unit discards the challenge response due to "Wrong SID". Based on observations in the previous chapter, this is expected, as the replayed message from the adversary does not use the correct randomly generated challenge SID which was included in the challenge request. This experiment also

confirmed that the control unit actually validates the challenge response and rejects messages with the wrong challenge SID.

Challenge-response message exchanges happen infrequent. The challenge request contains 7 bytes which are randomly chosen, resulting in a total number of over $2^{56} = 72\,057\,594\,037\,927\,936$ possible challenge request messages. An adversary will not be able to collect correct challenge responses for each possible challenge request.

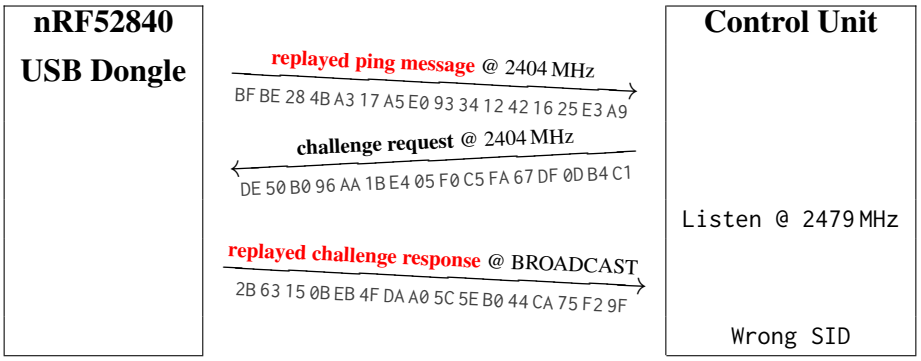


Figure 23: Replay attack for challenge responses

4.2.3 Replaying challenge requests

Resending a previously captured challenge request to the personal transceiver might not be directly useful for an adversary, but it can reveal additional information about the authentication protocol. First, a genuine authentication flow between the personal transceiver and the control unit using a logic analyzer connected to the SPI bus on the personal transceiver as shown in Section 3.3.1 is captured. Afterwards, the control unit will be disabled and the same captured challenge request is sent in response to the ping message using a nRF52840 USB dongle.

Figure 24 shows the messages exchanged during the challenge request replay attack. This attack can be performed successfully, the personal transceiver accepts the challenge request as if it is a genuine challenge request. This means that the content of the challenge request does not have any dependency on the ping message's randomness and the personal transceiver will not validate the freshness, allowing the replay of the challenge request. During the captured genuine authentication flow with the control unit, the personal transceiver switched to radio channel 2439 MHz after the challenge request. When replaying the same challenge request later, the personal transceiver switches again to radio channel 2439 MHz. This confirms the previously discovered fact, that the radio channel for the challenge response depends on the challenge request.

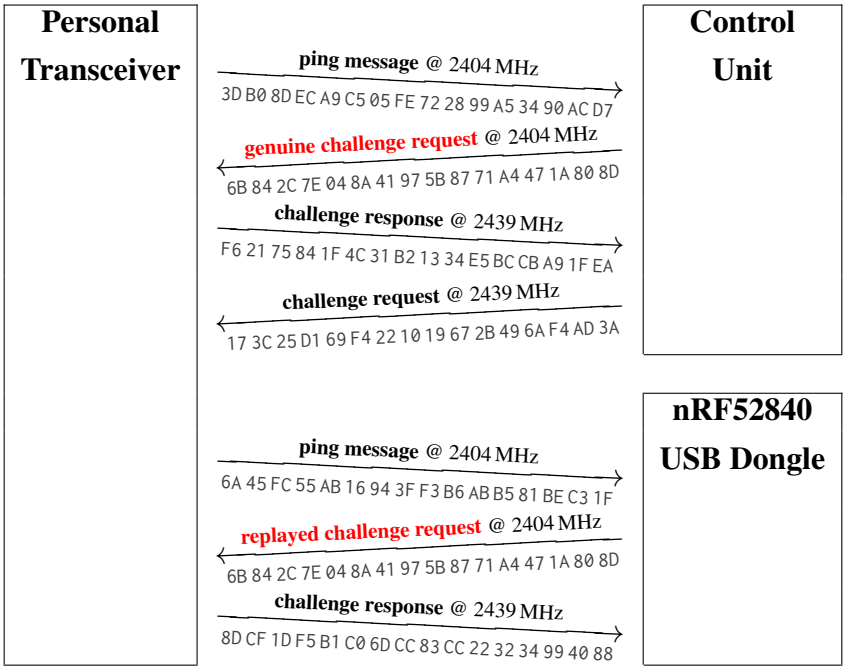


Figure 24: Replay attack for challenge requests

4.2.4 Replaying ping messages from different immobilizer set

Different immobilizer sets seem to use different logical addresses and exchange ping messages on different radio channels, such that the control unit will not pick up ping messages from different immobilizer sets. Using the nRF52840 USB dongle, it is possible to receive ping messages from one immobilizer set, change the logical address and radio channel, and forward them to the control unit of a different car immobilizer set. Figure 25 shows the message flow of the described attack scenario and the captured messages.

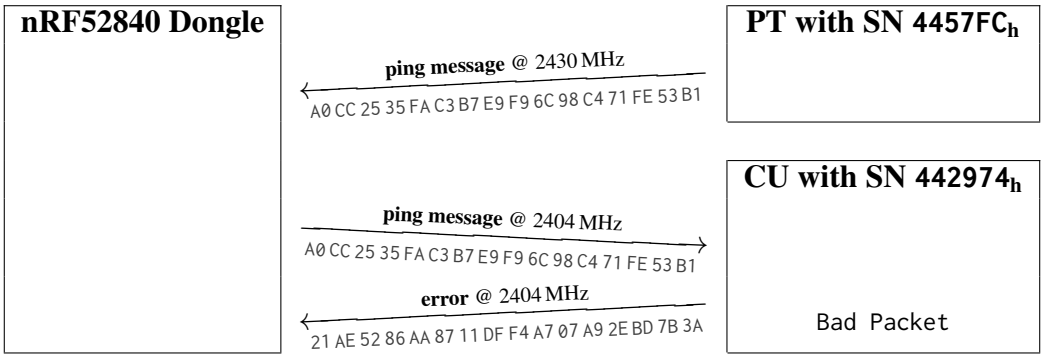


Figure 25: Replay attack for challenge requests

When sending ping messages from a different immobilizer set with different serial number to the control unit, it will reject the ping message and not respond with a challenge request. The reason "Bad Request" is logged on the UART debug console and an error message is returned on the same channel. It was discovered in previous sections that the service code is used as an encryption key for the AES-128 encryption, which means that immobilizer sets with different serial numbers (and service code) will not be able to decode each other's messages. Decrypting the AES-128 ciphertext with a wrong key will result in random looking messages which do not follow the correct message format. Therefore an adversary is not able to bypass the immobilizer by relaying messages from and to a personal transceiver belonging to a different immobilizer set.

4.3 Injection of errors

Injecting errors, like flipping bits of the exchanged messages, can reveal additional system properties. For this purpose, arbitrarily bits were flipped in the ciphertext in order to observe the behaviour of the system. This analysis has been performed for following messages:

Ping messages: If any bit of a ping message is altered, the control unit will reject the ping message with "Bad Packet" and respond with an error message.

Challenge request: If any bit of a challenge request is altered, the personal transceiver will simply ignore the message.

Challenge response: If any bit of a challenge response is altered, the control unit will reject the message with "Bad Packet" error.

Any flipped bit within an AES-128 ciphertext message will either result in invalid structure of the plaintext after decryption. The behaviour is consistent with previous observation, given that it was already established that all messages are encrypted with AES-128. Because AES-128 is not malleable when used to encrypt individual blocks in EBC mode, an attacker is unable to inject errors into the plaintext of the message without knowing the service code of the immobilizer, i.e. the actual encryption key.

4.4 Denial of service attack

This section explores attacks interfering with the immobilizer's radio communication, preventing successful authentication between control unit and personal transceiver. Rather than bypassing the car immobilizer, such attacks will immobilize the car, leaving the victim with an inoperable car that cannot be driven.

4.4.1 Radio jamming

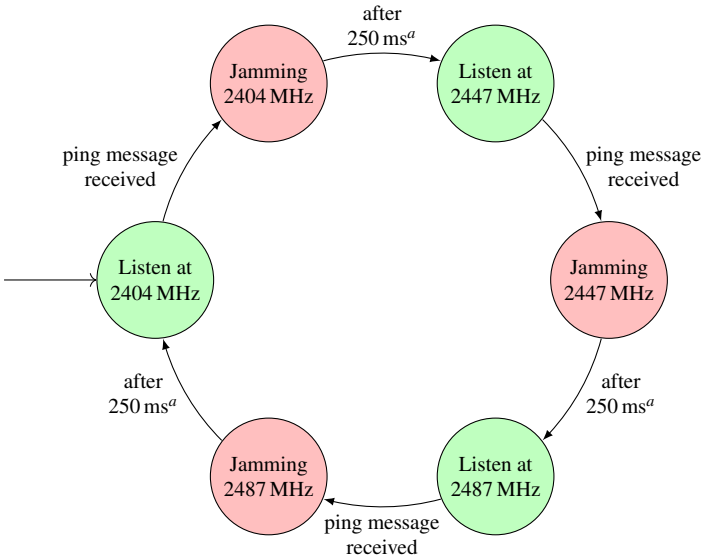
Bluetooth uses Gaussian frequency-shift keying (GFSK) modulation and has a similar frequency range and packet structure than the protocol used by the nRF24L01+ transceiver chip. Bluetooth also makes use of a channel hopping to increase its reliability against interferences and jamming attacks, but is still considered vulnerable to various jamming attacks [25]. Due to the similarities to the proprietary radio protocol used by the nRF24L01+ transceiver chip, it is likely that it is as well vulnerable to jamming attacks. Channel hopping cannot defend against high-power jamming attacks which span over the whole used frequency range, i.e. covers all available channels. Channel hopping can only defend against narrow-band low-power jamming attacks if an adversary cannot predict the selected channel sequence.

The Skybrake DD5 immobilizer exchanges the challenge requests and challenge responses on seemingly random channels whose selection algorithm is unknown. The car immobilizer therefore has sufficient protection against narrow-band low-power jamming attack once the control unit has completed the authentication with the personal transceiver. The main weakness of the Skybrake DD5 immobilizer is the initial message exchange in it is unauthenticated state. The immobilizer selects one out of three radio channels for the initial ping message exchange by using a predictable round-robin algorithm. These three radio channels are observable and the round-robin algorithm is known and deterministic, which makes narrow-band low-power jamming attacks very efficient.

The author of this work was able to perform a reliable jamming attack with an unauthenticated control unit using a single nRF52840 USB dongle. More precisely, a reactive jamming attack was performed in which jamming is only activated for a short period of time once a valid message has been received on the active radio channel. For this reason, the logical address of the immobilizer set needs to be prior known or captured using a software defined radio. The jamming effect was achieved by sending random messages of 16 bytes with the same logical address to the radio channel. This effectively prevented the personal transceiver from receiving the challenge request from the control unit after the ping message was sent. This operation is performed round-robin on the same channel sequence which is used by the car immobilizer. Assuming there is only one personal transceiver for the car immobilizer in the vicinity of the adversary, the described attack will automatically synchronize and follow the same channel hopping as performed by the personal transceiver. Figure 26 shows the state diagram for the reactive jamming attack for the car immobilizer model 520120 with serial number 442974_{h}.

Alternatively to the reactive jamming attack described here, it is possible to use three nRF52840 USB dongles and continuously jam all three channels on which ping messages are sent to.

Note that a user impacted by a radio jamming attack can mitigate this attack and be able



[a]Sufficient long to jam attempts of control unit to send the challenge request

Figure 26: State diagram of reactive jamming attack

to operate the car by temporarily disabling the immobilizer using the PIN code printed on the service card.

4.4.2 Deauthentication attack

As seen in the previous section, Skybrake DD5 immobilizer is vulnerable to low-power narrow-band radio jamming attacks during its initial authentication phase. To have a reliable denial of service attack after authentication already has occurred, an adversary needs to be able to *de-authenticate* an already authenticated car immobilizer, forcing it to perform the initial authentication phase again.

During experiments performed in Section 4.3 and Section 4.2.2, it was observed that the control unit will always enter the unauthenticated state once it receives any malformed challenge response. This effectively de-authenticates the personal transceiver.

In the authenticated state, the control unit spends the vast majority of the time listening for challenge responses from the personal transceiver. Broadcasting one random message with length of 16 bytes to all available 126 radio channels will with high probability be picked up by the control unit and be mistaken for the challenge response from the personal transceiver. This is a simple and efficient attack which will de-authenticate a car immobilizer with known logical address in the vicinity. For the deauthentication, the adversary needs to know the logical address of the immobilizer set, which can be captured using a software defined radio if needed.

Deauthentication will not immediately result in immobilization of the car. Prior to any immobilization, the control unit will enter a warning state and inform the driver about the immenent immobilization. An immobilization can only be forced by an adversary if an adversary uses the reactive jamming attack described in Section 4.4.1 after the deauthentication attack.

4.5 Relay station attacks

Relay station attacks can be performed by adversaries if they are in radio range of the personal transceiver. Common scenarios include personal transceivers placed near the entrance of a building where the signal can be captured from outside of the building, or adversaries which follow the victim carrying the personal transceiver on their person. An adversary can extend the range of the signal of the personal transceiver by relaying the signal over an alternative channel with increased range. For this, relay stations will be placed within the car and nearby the personal transceiver. On both relay stations, the signals will be captured, transmitted over the alternative channel, and transmitted by the other relay station. Figure 27 shows a schema of how an adversary can perform a relay attack by following the victim.

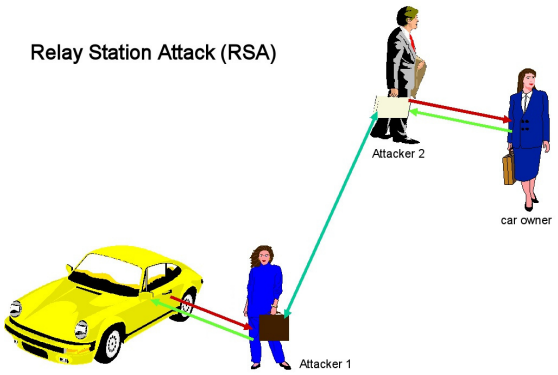


Figure 27: Relay station attack [26]

Because each signal is captured, relayed and re-transmitted, communication latency between the personal transceiver and the control unit will inevitably increase. In order to be resilient to relay attacks, the personal transceiver and control unit would need to be very sensitive to increased latency and prevent authentication in such situations.

We already have established that the control unit waits up to 3 seconds for any challenge responses, including the very first challenge response which will authenticate the personal transceiver. This is sufficient time to capture the challenge response from the personal transceivers, transfer it over the alternative channel, and transmit it to the control unit.

Based on the captured SPI traffic on the personal transceiver, it is known that the personal transceiver does only wait for as little as 2 ms in order to receive the challenge request after sending the ping message. Based on the estimations done in Section 4.2.2, it is known that the on-air time of the ping message and the challenge request together will be around 286 µs. Even with all the ramp-up times of the nRF24L01+ radio transceivers, it is likely that up to 1 ms is left as an additional buffer. Modern relay station attacks introduce as little as 15-20 µs additional latency to relayed signals [27]. Channel hopping does not prevent the relay station attack for the first time-sensitive challenge request, as it is sent to the identical handshake channel as the ping message was sent to. Although the author of this work has not attempted such relay station attacks against the Skybrake DD5 immobilizer, it seems likely that the immobilizer is vulnerable to relay station attacks.

4.6 Privacy implications

Personal transceivers of the Skybrake DD5 car immobilizer will continuously announce themselves by sending ping messages. The ping messages have fixed-length, valid checksums and are sent in a recognizable pattern on three different radio channels. Using a software defined radio (SDR) with sufficient bandwidth, an adversary can detect if someone is carrying a Skybrake DD5 immobilizer. Laks [5] has successfully implemented such an attack for the predecessor Skybrake DD2 immobilizer without SDR solely relying on the preamble, logical address prefix and the valid CRC checksums. Similar attacks against the Skybrake DD5 require a SDR due to the different packet structure and would need to detect valid Skybrake DD5 packets based on the preamble, packet length field and valid CRC checksum.

Ping packets transmited by Skybrake DD5 immobilizers are following a known sending pattern which can help to recognize packets sent by the immobilizers. The logical address as well as the radio channels over which they are transmitted allow distinguishing between different immobilizer sets. Once the logical address and radio channels of the immobilizer set are known, an adversary can perform the tracking using simple nRF52840 USB dongles instead of using a SDR. Users who carry the personal transceivers on their person can therefore be individually tracked. Their presence or absence can be detected and their precise location within a room might be obtained through signal strength triangulation.

5 Firmware and memory dumping

Previous sections cover black-box analysis where only input and output of immobilizer were observed, without having detailed knowledge of the microcontroller internals. While this limited approach provided many clues how the immobilizer works, it is not sufficient to fully reconstruct the working principle of all features supported by the car immobilizer. Having access to the actual program code of the immobilizer would allow for more in-depth analysis of the working principle by disassembling and reverse-engineering the individual instructions executed by the microcontrollers.

The PIC24F family microcontrollers used in the car immobilizer are executing the firmware code from its own memory located within the microcontroller chip package. This chapter focuses on the extraction of the memory contents of the car immobilizer and its programming device in order to gain access to the program code for a more in-depth analysis.

5.1 Microcontroller configuration

The manufacturer did expose the in-circuit serial programming (ICSP) of the control unit and the personal transceiver through test pads. It is likely that the manufacturer Autonams did not order pre-programmed chips from Microchip [28] and was required to expose the ICSP interface to program the microcontrollers after the PCB assembly with the correct firmware and configuration data.

Using the exposed ICSP interface, the configuration registers of the PIC microcontroller can be read using a PICkit 3 [29] with the MPLAB Integrated Programming Environment (IPE) [30] software. The microcontroller configuration of the control unit and the personal transceiver was successfully read and are available in Appendix IV. From security perspective, it is important to note, that in order to prevent extracting of the memory contents containing the program code through the programming interface, the manufacturer has enabled the standard code protection feature of the microcontroller. Writing operations to the microcontroller are only possible after a full chip erase has been performed, which would loose any data of interest.

5.2 Firmware extraction

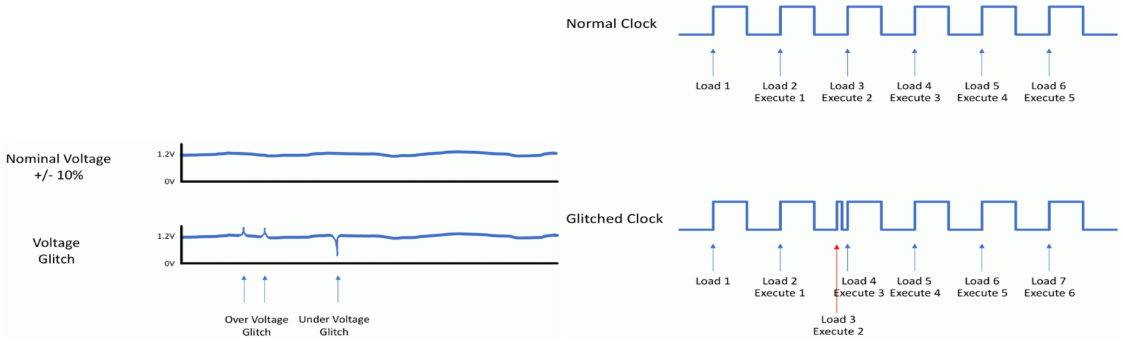
Microcontrollers combine the processor and memory within a single chip package. When code protection features are enabled on the microcontroller, the programming interface exposed by the chip prevents access to the internal memory containing the program code. However, an adversary who has physical access to the microcontroller has a wide variarity of possible invasive and non-invasive attacks he can perform in order to bypass this protection and gain access to the data stored within the memory of the

microcontroller [31]. The manufacturer of the microcontroller has acknowledged that even with the code protection feature enabled, no manufacturer can guarantee that the memory content remain secret:

> Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable." [11]

To the knowledge of the author of this work, there are no publicly documented methods which would bypass the code protection feature of the PIC24F microcontroller series from Microchip used in the immobilizer as of August 2024.

5.2.1 Glitching attacks



(a) Voltage Glitching (b) Clock Glitching

Figure 28: Glitching attacks to introduce faults into microcontrollers [32]

Glitching refers to a type of malicious input that intentionally induces errors or anomalies (glitches) in the processor's behavior. This can be done by carefully crafting under-voltage or over-voltage spikes on the power supply as shown in Figure 28a, or clock pulses which are outside of the specification of the microcontroller as shown in Figure 28b.

Glitching disrupts the normal flow of instructions and data processing. The goal is to create an unpredictable behavior that can lead to:

Data corruption Data being accessed in registers and memory might be corrupted

Unintended code execution Unintended code branches might be executed because some instructions are completely skipped (like jump instructions) or conditional statements in the control flow are wrongly evaluated (like compare instructions).

The author of this work attempted to bypass the code protection feature of a PIC24F16KA101 microcontroller using power supply voltage glitching attacks. A ChipWhisperer-Lite

board [33] was used for injecting power supply glitches while simultaneously performing read operations through the programming interface.

Unfortunately, the author of this work was not able to find any configuration of glitch duration and glitch timing which would allow bypassing the code protection. However, this does not mean that the microcontroller is not vulnerable to such attacks. It might just be that correct parameters for the glitch duration and glitch timing were not found, or that the microchip requires multiple glitch operations in order to inject the desired fault and be able to bypass the code protection.

5.2.2 Firmware dumping as a service

Another type of attacks are to bypass the packaging of the chip, i.e. the physical security boundary, in order to gain access to the embedded silicon die contaning the memory. However, invasive attacks which expose the silicon die contained within the microcontroller to directly access the memory are requiring expensive equipment. They are prone to errors and require experience and prior research in order to execute them reliably.

There are service providers[5] such as [34], [35], [36] and [37] which specialize in reverse engineering of microcontrollers. They have dedicated research facilities where they perform research on different microcontrollers to work out working attacks against particular microcontroller models. Those service providers offer extraction firmware from microcontrollers as a service.

While there is no publicly known vulnerability for the microcontrollers used within the immobilizer, the service provider RussianSemiResearch [34] does claim on their website to be able to extract firmware from PIC24F16KA101 and PIC24F32KA302 microcontrollers.

[5]Please note that the author of this has not verified the claims of the service providers listed in this section and does not endorse them.

6 Conclusion

Previous chapters described the reverse engineering process of the Skybrake DD5 immobilizer and explored various possible attack vectors. This chapter aims to provide a synthesis of the major findings and identifies areas where additional future work is required in order to fully assess the security properties of the Skybrake DD5 immobilizer.

6.1 Discussion

Through black-box analysis, it was possible to gather information into the working principle of the Skybrake DD5 immobilizer. It was confirmed that the immobilizer uses AES-128 for all exchanged messages. Each immobilizer uses their own encryption key, which is encoded in the service code printed on the service card provided to the end customer. A possible improvement, albeit security through obscurity, might be to derive the actual AES-128 encryption key using the service code and some other static secret which is protected within the microcontrollers.

The Skybrake DD5 immobilizer uses a challenge-response protocol. The initial authentication is initiated by the personal transceiver by regularly sending ping messages. If the car ignition is switched on, the control unit will respond to received ping messages with a challenge to the personal transceiver. For successful authentication, the personal transceiver needs to send back the correct response.

We identified key fields contained within the message plaintext of the exchanged messages. Even though AES-128 is a deterministic block cipher, all plaintext messages contain at least three random bytes which makes the ciphertext look random. The ping message contains the serial number and an ID of the personal transceiver. The challenge request includes a randomly generated 32-bit challenge SID and the radio channel on which the response for the challenge is expected on. The challenge for the personal transceiver is to decrypt the challenge request to gain access to the challenge SID, and re-encrypt a challenge response containing the same challenge SID. In addition with each challenge response, the personal transceiver also sends its battery voltage.

The service code printed on the service card of the immobilizer is used as an AES-128 encryption key. If the service code, serial number and logical address is known, the functionality of the personal transceiver required during authentication can be reimplemented and the car immobilizer can be effectively bypassed. Although not clearly stated in any user manual, it is therefore crucial that users do not reveal their service code to potential adversaries. As an improvement, it might be good to hide the service code under a scratch pad, as it was done with the PIN code. At the very least, this would indicate to users that the service code is sensitive data which needs to be protected.

Some types of replay attacks against the Skybrake DD5 immobilizer were successful, but

a full replay attack against the control unit was prevented through the randomly generated challenge. Upon receiving challenge responses, the immobilizer correctly validates the challenge responses and rejects replayed messages when a mismatch of the challenge has been detected.

As with many radio connected devices, the Skybrake DD5 immobilizer is susceptible to high-power jamming attacks of the whole frequency range. It was possible to show a more efficient denial of service attack using de-authentication and narrow-band low-power jamming.

6.2 Future work

The black-box reverse engineering of the DD5 immobilizer was able to provide a high-level overview of the security protocol used by the car immobilizer. A more complete picture could be obtained if the firmware could be extracted from the immobilizer device. Disassembling and reverse engineering the firmware would allow to fully document the protocol and make an in-depth analysis of the security properties involving all so far unknown supported message types. Doing black-box analysis of Skybrake DDWR5 programming device and dumping the memory of its microcontroller might give further insights into how the Skybrake DD5 immobilizer works and might reveal additional security flaws that might be hidden in custom supported messages.

It is unclear how the immobilizer derives its logical address and the three handshake channels used to transmit the ping messages. The Skybrake DDWR5 programming device likely needs to derive this information from either the serial number or the service code, but its unclear how this is done. Understanding how these parameters are derived might improve some of the described attacks as the adversary would then need less information. In particular, it is unclear if the handshake channels are different amongst immobilizer sharing the same model number and firmware version.

The functionality of the personal transceiver required for successful authentication can be fully reimplemented on a nRF52840 USB dongle, given the service code, serial number and logical address is known. Such a re-implementation could be used to confirm that the working principle is fully understood and applies to immobilizers of different versions. It could also be useful for users who have lost or damaged their original personal transceiver, or for users who are in need of additional personal transceivers.

References

[1] Regulation (EU) 2019/2144 on type-approval requirements for motor vehicles, November 2019. URL http://data.europa.eu/eli/reg/2019/2144/2024-07-07. Last accessed 13 August 2024.

[2] ManaPolise. Conditions for Kasko insurance (in Latvian). URL https://www.manapolise.lv/lv/par-kasko-polisi/. Last accessed 13 August 2024.

[3] BALTA. Vehicle theft trends in Latvia (in Latvian). URL https://www.balta.lv/lv/uznemumiem/kasko/transportlidzeklu-zadzibu-tendences-latvija. Last accessed 13 August 2024.

[4] Craig Smith. *The Car Hacker's Handbook: A Guide for the Penetration Tester*, pages 215–231. No Starch Press, 2006.

[5] Jürgen Laks. Reverse Engineering a Car Immobilizer. Master's thesis, University of Tartu, 2023.

[6] Autonams. "Car immobilizers (in Latvian)" archived page, 14 July 2013. URL https://web.archive.org/web/20130714001640/http://www.autonams.lv:80/auto-drosiba/auto-imobilaizeri/. Last accessed 13 August 2024.

[7] Autonams. Car immobilizers. URL https://www.autonams.lv/en/car-immobilisers_/. Last accessed 13 August 2024.

[8] *Skybrake DD5 User Manual*. Skybrake, February 2013. URL http://discoverloop.com/file/repository/Skybrake_DD5_USER_MANUAL_EN.pdf. Last accessed 13 August 2024.

[9] *Skybrake DD5 Installation Guide*. Skybrake, February 2013. URL https://alfapolish.hu/letolt/skybrake_dd5.pdf. Last accessed 13 August 2024.

[10] Microchip. PIC24FV32KA304 Family - Data Sheet, November 2017. URL https://ww1.microchip.com/downloads/en/DeviceDoc/30009995e.pdf. Last accessed 13 August 2024.

[11] Microchip. PIC24F16KA102 Family - Data Sheet, November 2011. URL https://ww1.microchip.com/downloads/en/DeviceDoc/39927c.pdf. Last accessed 13 August 2024.

[12] Minimal dumb-terminal emulator picocom, 12 April 2018. URL https://github.com/npat-efault/picocom. Last accessed 13 August 2024.

[13] eng.auto24.ee. "Skybrake DD5" product listing. URL https://eng.auto24.ee/products/product_pic.php?id=1149889&view=20. Last accessed 13 August 2024.

[14] *nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification v1.0*. Nordic Semiconductor, April 2010. URL https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf. Last accessed 13 August 2024.

[15] gr-nrf24-sniffer tool to receive and decode wireless traffic from nRF24L01(+), 31 October 2022. URL https://github.com/kittennbfive/gr-nrf24-sniffer. Last accessed 13 August 2024.

[16] www.bitcraze.io. Sniffing nRF24 with GNU Radio and HackRF. URL https://www.bitcraze.io/documentation/tutorials/hackrf-nrf/. Last accessed 13 August 2024.

[17] Great Scott Gadgets. HackRF One. URL https://greatscottgadgets.com/hackrf/one/. Last accessed 13 August 2024.

[18] sigrok.org. sigrock signal analysis software suite. URL https://sigrok.org/. Last accessed 13 August 2024.

[19] *nRF52840 Specification v1.8*. Nordic Semiconductor, December 2023. URL https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.8.pdf. Last accessed 13 August 2024.

[20] John Walker. Pseudorandom number sequence test program, January 2008. URL https://www.fourmilab.ch/random/. Last accessed 13 August 2024.

[21] Donald E. Knuth. *The Art of Computer Programming*. Addison Wesley, 2. edition, 1969.

[22] auto.bazos.sk. "Skybrake DD5" product listing (in Slovak). URL https://auto.bazos.sk/inzerat/167659129/skybrake-dd5.php. Last accessed 13 August 2024.

[23] elmir.ua. "Skybrake DD5" product listing (in Russian). URL https://elmir.ua/immobilizer/immobilizer_skybrake_dd5_5211.html. Last accessed 13 August 2024.

[24] alarmtuning.cz. "Skybrake DD5+" product listing (in Czech), . URL https://alarmtuning.cz/bezkontaktni-imobilizer-1-okruhovy-skybrake-sk-dd5.htm. Last accessed 13 August 2024.

[25] Hossein Pirayesh and Huacheng Zeng. Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 24(2):767–809, 2022. doi: 10.1109/COMST.2022.3159185.

[26] Wikimedia Commons. Relay Station ATtack, August 2006. URL https://commons.wikimedia.org/wiki/File:RelayStationAttack.jpg. Last accessed 13 August 2024.

[27] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. January 2011.

[28] Microchip. Programming Services, . URL https://www.microchip.com/en-us/microchipdirect/programming-services. Last accessed 13 August 2024.

[29] Microchip. PICkit™ 3 In-Circuit Debugger, . URL https://www.microchip.com/en-us/development-tool/pg164130. Last accessed 13 August 2024.

[30] Microchip. MPLAB® Integrated Programming Environment (IPE), . URL https://microchip.com/en-us/tools-resources/production/mplab-integrated-programming-environment. Last accessed 13 August 2024.

[31] Sergei P. Skorobogatov. Semi-invasive attacks – A new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005. URL https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf.

[32] Bart Stevens. Fault Injection Attacks: A Growing Plague, March 2019. URL https://www.eeweb.com/fault-injection-attacks-a-growing-plague/. Last accessed 13 August 2024.

[33] NewAE Technology Inc. ChipWhisperer-Lite 32-Bit. URL https://www.newae.com/products/nae-cwlite-arm. Last accessed 13 August 2024.

[34] RussianSemiResearch. Read protected Microcontrollers (MCU). URL https://russiansemiresearch.com/en/service/#read_prot. Last accessed 13 August 2024.

[35] www.ic-cracker.com. MCU Crack Category. URL https://www.ic-cracker.com/category/mcu-crack/. Last accessed 08 August 2024.

[36] www.crackyouric.com. Crack Your IC. URL https://www.crackyouric.com/. Last accessed 13 August 2024.

[37] www.break-ic.com. Microcontroller reverse engineer. URL https://www.break-ic.com/. Last accessed 13 August 2024.

[38] alarmtuning.cz. "Skybrake Programmer DDWR5" product listing, . URL https://alarmtuning.cz/programator-bezdratovy-pro-sk-dd5-nebo-sk-dd5-motion-skybrake-sk-ddwr5.htm. Last accessed 13 August 2024.

[39] *SkyBrake DD5+ Motion Immobilizer with Wireless Control Installation Instruction*. Skybrake, February 2013. URL https://www.speedtech.cz/files/products/8055/imobilizer-bezdotykovy-sk-dd5-motion.pdf. Last accessed 13 August 2024.

[40] *AT91 ARM® Thumb®-based Microcontrollers - AT91SAM7S64*. Atmel, May 2005. URL https://www.keil.com/dd/docs/datashts/atmel/at91sam7s64_ds.pdf. Last accessed 13 August 2024.

[41] *64-megabit 2.7-volt Dual-interface DataFlash® - AT45DB642D*. Atmel, May 2004. URL https://datasheet.octopart.com/AT45DB642D-CNU-Atmel-datasheet-9652374.pdf. Last accessed 13 August 2024.

[42] Adafruit. FT232H Breakout - General Purpose USB to GPIO, SPI, I2C. URL https://www.adafruit.com/product/2264. Last accessed 13 August 2024.

[43] www.flashrom.org. flashrom utility for identifying, reading, writing, verifying and erasing flash chips. URL https://www.flashrom.org/. Last accessed 13 August 2024.

Appendices

I Programming Device

In addition to the two car immobilizer sets, the author of this work also had access to a programming device (see Figure 29) for the Skybrake DD5 immobilizer. These programmers are usually only available for authorized service partners who install the immobilizer and are not distributed to the general public. The programmer device allows to change configuration of the control unit in the car and allows pairing additional personal transceivers with the control unit.

Figure 29: Programmer device DDWR5 [38]

This section will briefly introduce the programming device and its components. Note that the author of this work did not perform an in-depth analysis of the programming device. Neither the software was reverse engineered nor were the messages sent and received by the programming device analyzed.

Software

Authorized service partners will be provided with an Immobilizer Management Software (IMS) which can be used to configure the immobilizer after installation or adjust the immobilizer settings if needed. The information in this section is based on the user manual [39] which is available to authorized service partners.

In order to establish a connection with the immobilizer using the Immobilizer Managment Software, the service partner is required to enter the serial number and service code (see Figure 30) which is written on the service card of the immobilizer. To successfully communicate with the immobilizer, the programmer device is required to configure the nRF24L01+ transceiver chip with the matching logical address and a valid radio channel. This makes it likely that the logical address and the radio channels for the control unit can be derived from serial number and the service code of the immobilizer.

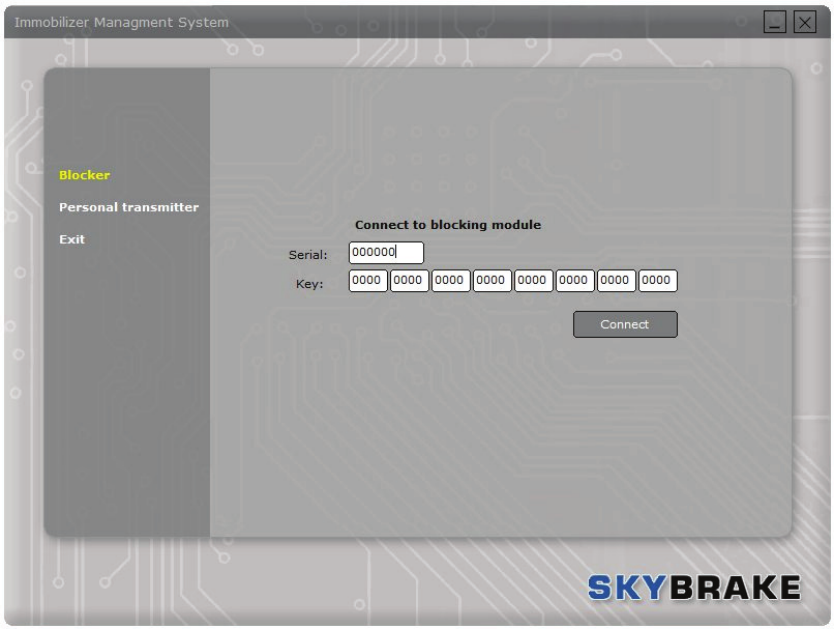
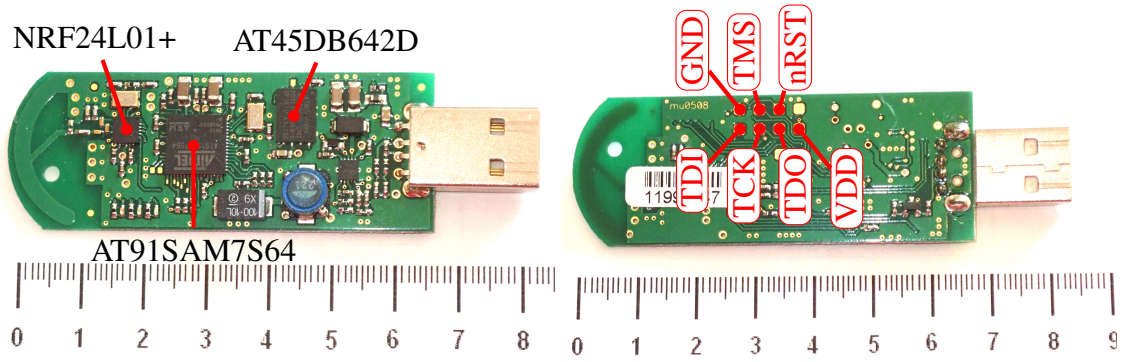


Figure 30: Establishing connection using the IMS software [39]

Internal Components

Figure 31a shows the programmer with its main components. It is built on a different microcontroller architecture than the control unit or the personal transceiver. It uses a AT91SAM7S64 ARM microcontroller by Atmel running at 55 MHz. The microcontroller has 64 KiB flash memory to hold program code and 16 KiB SRAM [40]. Additional 8 MiB storage is made available to the microcontroller through an AT45DB642D flash memory chip by Renesas [41], which is connected over an serial peripheral interface (SPI) bus. The nRF24L01+ transceiver chip over an SPI bus is used for communicating with the control unit and personal transceiver.

Figure 31b shows the back of the PCB which contains test pads exposing the JTAG interface of the microcontroller.



(a) Front of PCB of the programmer device (b) Back of PCB of the programmer device

Figure 31: Exposed PCB of the programmer device

Memory dump

Unlike the memory of microcontrollers which is embedded within the microcontroller and offers code protection features, the dedicated AT45DB642D flash memory holding auxiliary data can be dumped by using for example an Adafruit FT232H Breakout board [42] with `flashrom` [43]. A dump of the memory chip was successfully obtained and it was discovered that it contains only 1 056 bytes of seemingly random data, while the rest of the 8 MiB flash memory chip remains empty. It is likely that the dedicated memory chip is used for storing data which increases and accumulates over time during its operation and the programming device was simply little used.

The author of this work was not able to dump the memory of the AT91SAM7S64 microcontroller containing the actual firmware of the programming device, as the microcontroller has code protection features enabled.

II UART debug output during startup

Listing 2 and Listing 3 shows the data that has been captured from the UART serial debug interface during the startup sequence of the car immobilizers.

```
DD5 Master Blocker
==========================================
[5]DD5 Serial Number 0x00442974
[8]Reading EEPROM...
[11]Conf #A valid
[13]Config loaded in 5 [ms]
[15]Voltage metering module OK
[18]IO module init OK
[170]VM adc=346 voltage=12.1 [V]
[173]IO Relay duty = 99 [%]; Buzzer duty = 49 [%]
SW Version 22
Build: Dec 18 2015 - 01:13:48
```

Listing 2: Startup debug output of model 520120 with serial number 442974_h

```
DD5 MBLO
==========
[0.002]VM OK
[0.003]VM adc=83 voltage=4.7 [V]
[0.007]Relay duty = 100 %
[0.010]DD5 s/n 0x004457FC
[0.013]Reading EEPROM..
[0.018]Conf #A ok
[0.021]Conf loaded in 9 ms
[0.024]IO OK
[0.175]VM adc=267 voltage=12.5 [V]
[0.179]Relay duty = 100 %
SW V 33
Jan 26 2018 - 15:04:25
MCU: 4512 HW Rev: 7
```

Listing 3: Startup debug output of model 520160 with serial number $4457FC_h$

III UART debug output during authentication flow

Listing 4 shows the data that has been captured from the UART serial debug interface when the control unit performs the authentication with the personal transceiver.

```
[102646]DD5 nRF IRQ
[102663]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
[102666]Rx @ ch. 87
[102669]PT Ping Packet
[102671]Li @ 92 next (99)
[102673]Wait 3506 ms

[102848]DD5 nRF IRQ
[102865]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
[102868]Rx @ ch. 92
[102871]PT BAT: OK (2987)
[102873]LinkQ: 0
[102874]Ping -> USB
[102963]Config update in 28 [ms]
[102966]PT Cmd A:1 R:1
[102968]Li @ 99 next (84)
[102971]Wait 3556 ms

[102973]RELAY: UNBLOCK
[102992]Config update in 17 [ms]
[102994]SIG: 0
[102996]PT OK
[102997]Anti Hijack mode
[106018]DD5 nRF IRQ
[106035]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
[106038]Rx @ ch. 99
[106041]PT BAT: OK (2987)
[106043]LinkQ: 0
[106044]Ping -> USB
[106106]PT Cmd A:1 R:2
[106109]Li @ 84 next (116)
[106111]Wait 3531 ms

[109388]DD5 nRF IRQ
[109404]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
[109407]Rx @ ch. 84
[109410]PT BAT: OK (2987)
```

```
38 [109412]LinkQ: 0
39 [109413]Ping -> USB
40 [109475]PT Cmd A:1 R:3
41 [109478]Li @ 116 next (106)
42 [109480]Wait 3555 ms
43
44 [112558]DD5 nRF IRQ
45 [112575]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
46 [112578]Rx @ ch. 116
47 [112581]PT BAT: OK (2988)
48 [112583]LinkQ: 0
49 [112584]Ping -> USB
50 [112646]PT Cmd A:1 R:4
51 [112649]Li @ 106 next (90)
52 [112651]Wait 3558 ms
53
54 [115728]DD5 nRF IRQ
55 [115744]TxFIFO(1/0), RxFIFO(0/0), STATUS 65
56 [115747]Rx @ ch. 106
57 [115750]PT BAT: OK (2989)
58 [115752]LinkQ: 0
59 [115753]PT Cmd A:1 R:5
60 [115756]Li @ 90 next (33)
61 [115758]Wait 3504 ms
```

Listing 4: Authentication debug output of model 520160 with serial number $4457FC_h$

IV Microcontroller configuration

PIC24F32KA302 of control unit

Following microcontroller configuration has been read from the control unit of immobilizer model 520120 with serial number 442974_h.

Address	Name	Field	Value	Option	Setting
$F80000_h$	FBS	BWRP	E_h	ON	Boot Segment Write Protect Enabled
		BSS	7_h	OFF	No boot program flash segment
$F80004_h$	FGS	GWRP	1_h	OFF	General segment may be written
		GSS0	0_h	ON	Standard security enabled
$F80006_h$	FOSCSEL	FNOSC	22_h	PRI	Primary Oscillator (XT, HS, EC)
		SOSCSRC	1_h	ANA	Analog Mode for use with crystal
		LPRCSEL	0_h	LP	Low Power, Low Accuracy Mode
		IESO	0_h	OFF	Internal External Switchover mode disabled (Two-speed Start-up disabled)
$F80008_h$	FOSC	POSCMOD	$3A_h$	HS	HS oscillator mode selected
		OSCIOFNC	0_h	OFF	CLKO output disabled
		POSCFREQ	3_h	HS	Primary oscillator/external clock input frequency greater than 8MHz
		SOSCSEL	1_h	SOSCHP	Secondary Oscillator configured for high-power operation
		FCKSM	0_h	CSECME	Both Clock Switching and Fail-safe Clock Monitor are enabled
$F8000A_h$	FWDT	WDTPS	$4F_h$	PS32768	Watchdog Timer Postscale Select bits 1:32768
		FWPSA	0_h	PR32	WDT prescaler ratio of 1:32
		FWDTEN	0_h	OFF	WDT disabled in hardware; SWDTEN bit disabled
		WINDIS	1_h	OFF	Standard WDT selected(windowed WDT disabled)
$F8000C_h$	FPOR	BOREN	$D6_h$	BOR2	Brown-out Reset enabled only while device is active and disabled in SLEEP, SBOREN bit disabled
		LVRCFG	1_h	OFF	Low Voltage regulator is not available
		PWRTEN	0_h	OFF	Power-up Timer disabled
		I2C1SEL	1_h	PRI	Use Default SCL1/SDA1 Pins For I2C1
		BORV	2_h	V27	Brown-out Reset set at 2.7V
		MCLRE	1_h	ON	RA5 input pin disabled,MCLR pin enabled
$F8000E_h$	FICD	ICS	$E3_h$	PGx1	EMUC/EMUD share PGC1/PGD1
$F80010_h$	FDS	DSWDTPS	$5F_h$	DSWDTPS	Deep Sleep Watchdog Timer Postscale Select bits 1:2,147,483,648 (25.7 Days)
		DSWDTOSC	1_h	LPRC	DSWDT uses Low Power RC Oscillator (LPRC)
		DSBOREN	1_h	ON	Deep Sleep Zero-Power BOR Enable bit Enabled

PIC24F16KA101 of personal transceiver

Following microcontroller configuration has been read from the personal transceiver of immobilizer model 520120 with serial number 442974_{h}.

Address	Name	Field	Value	Option	Setting
$F80000_{h}$	FBS	BWRP	E_{h}	ON	Boot segment is write-protected
		BSS	7_{h}	OFF	No boot program Flash segment
$F80004_{h}$	FGS	GWRP	1_{h}	OFF	General segment may be written
		GCP	0_{h}	ON	Standard security enabled
$F80006_{h}$	FOSCSEL	FNOSC	0_{h}	FRC	Fast RC oscillator (FRC)
		IESO	0_{h}	OFF	Internal External Switchover mode disabled (Two-Speed Start-up disabled)
$F80008_{h}$	FOSC	POSCMOD	93_{h}	NONE	Primary oscillator disabled
		OSCIOFNC	0_{h}	ON	CLKO output disabled; pin functions as port I/O
		POSCFREQ	2_{h}	MS	Primary oscillator/external clock input frequency between 100 kHz and 8 MHz
		SOSCSEL	0_{h}	SOSCLP	Secondary oscillator configured for low-power operation
		FCKSM	2_{h}	CSDCMD	Both Clock Switching and Fail-safe Clock Monitor are disabled
$F8000A_{h}$	FWDT	WDTPS	CF_{h}	PS32768	Watchdog Timer Postscale Select bits 1:32,768
		FWPSA	0_{h}	PR32	WDT prescaler ratio of 1:32
		WINDIS	1_{h}	OFF	Standard WDT selected; windowed WDT disabled
		FWDTEN	1_{h}	ON	Watchdog Timer enabled
$F8000C_{h}$	FPOR	BOREN	FA_{h}	BOR2	Brown-out Reset enabled only while device is active and disabled in Sleep; SBOREN bit disabled
		PWRTEN	1_{h}	ON	Power-up Timer enabled
		I2C1SEL	1_{h}	PRI	Default location for SCL1/SDA1 pins
		BORV	3_{h}	V18	Brown-out Reset set to lowest voltage (1.8V)
		MCLRE	1_{h}	ON	MCLR pin enabled; RA5 input pin disabled
$F8000E_{h}$	FICD	ICS	$E3_{h}$	PGx1	PGC1/PGD1 are used for programming and debugging the device
		BKBUG	1_{h}	OFF	Background debugger disabled
$F80010_{h}$	FDS	DSWDTPS	$5F_{h}$	DSWDTPS	Deep Sleep Watchdog Timer Postscale Select bits 1:2,147,483,648 (25.7 Days)
		DSWDTOSC	1_{h}	LPRC	DSWDT uses LPRC as reference clock
		RTCOSC	0_{h}	LPRC	RTCC uses LPRC as reference clock
		DSBOREN	1_{h}	ON	Deep Sleep BOR enabled in Deep Sleep

V Non-exclusive license to reproduce thesis and make thesis public

I, **Daniel Würsch**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive license) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Security Analysis of Skybrake DD5 immobilizer,
   *(*title of thesis)

   supervised by Arnis Paršovs.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Daniel Würsch
13 August 2024