Automatic Smart Key

Base Station Pairing for Automotive Keyless Go Systems

Master Thesis - Telematics

Philipp Miedl

Supervisor:
Dipl. Ing. Dr.techn. Franz Wotawa
Dipl. Ing. Bernd Janger

Institute for Software Technology - Graz University of Technology
and
Maxim Integrated - Lebring, Austria

Graz
April 1, 2014

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ………………………………………………..
date (signature)

Acknowledgment

I want to thank Prof. Franz Wotawa who supported me through the process of this project with valuable input and hints from an academic perspective. Also I want to express my gratitude to Maxim Integrated, especially Dipl. Ing. Bernd Janger and Dipl. Ing. Edwin Taferner, who gave me the chance, to do this thesis as part of the Maxim Integrated Keyless Go Project. Finally in this final work of my study, I want to thank my family and especially my parents who have supported me through all my years of study in every possible way.

Abstract

In today's automotive industry, electronics are getting more and more important in order to meet the customers demand on higher comfort and more functionalities in cars. One of those components, to increase the comfort, is the Keyless Go System. Those allow the driver to lock or unlock a car without interacting with a key, through button press or mechanically opening the lock. Most of those systems work with a Low Frequency (LF) and a Ultra High Frequency (UHF) (ISM-Band) link, between the Intelligent Key Unit used in the Keyless Go System (Key-Fob) and the car (Base Station). Advanced systems also offer a LF Transponder (TRP) link, to be able to lock, unlock and start the car over a wireless link even when the Key-Fob battery is empty.

As automotive manufacturers are facing big numbers of cars being build, it is highly important that all the components can easily be implemented and used. This requires a high level of automation when assembling. In case of Keyless Go Systems, an automatic pairing of the Key-Fob and the Base Station is needed. Automatic Smart Key implements such a automatic pairing procedure, which is later-on used in the context of the Maxim Integrated Keyless Go System (KGO System).

Moreover, Keyless Go Systems operate with more than one Key-Fob per car, therefore a technique needs to be found to handle the communication of multiple entities via one communication channel. In the context of this master thesis, such techniques as well as pairing strategies that are already used in industry are introduced and analysed. On basis of this analysis, an algorithm is developed which fits the needs of the KGO System perfectly. As this procedure needs to meet highest quality standards in the automotive industry, another point of focus is verification and test of the algorithm.

Kurzfassung

In der Automobilindustrie wird die Elektronik mittlerweilen immer wichtiger, um der Forderung der Kunden nach größtmöglichen Komfort und mehr Funktionalität des Autos nachzukommen. Eines dieser elektronischen Komponenten, zur Erhöhung des Komforts, sind die Keyless Go Systeme. Diese erlauben es dem Fahrer das Auto zu versperren oder entsperren, ohne den Schlüssel aktiv zu verwenden, weder durch Tastendruck noch durch bedienen des mechanischen Schlosses. Die meisten dieser Systeme verwenden Low Frequency (LF) und Ultra High Frequency (UHF) (ISM-Band) als Kommunikationskanäle zwischen dem intelligenten Schlüssel (Key-Fob) und dem Auto (Base Station). Manche der Systeme besitzen auch eine LF Transponder (TRP) Schnittstelle, um ein schlüsselloses Auf-, Absperren und Starten des Autos zu ermöglichen, selbst wenn die Batterie des Key-Fobs leer ist.

Da Automobilhersteller sich mit sehr hohen Stückzahlen konfrontiert sehen, ist es sehr wichtig, dass die einzelnen Komponenten leicht in das Endprodukt eingebaut werden können. Dies erfordert ein hohes Maß an Automatisierung bei der Zusammensetzung des Produkts, im Fall des Keyless Go Systems eine automatisierte Prozedur um die intelligenten Schlüssel mit dem Auto zu verbinden, zu Pairen. Ein Problem von Keyless Go Systemen ist auch, das diese mit mehreren Key-Fobs per Auto (Base Station) operieren, daher muss ein Weg gefunden werden um Kollisionen zu behandeln, da alle Key-Fobs mit der Basisstation über denselben Kanal kommunizieren.

Diese Diplomarbeit beschäftigt sich mit der mit der Analyse bestehender Kollisionsmanagament und Pairing Prozeduren. Aus den Erkenntnissen dieser Analyse wird eine solche automatisierte Pairing und Kollisionsmanagement Prozedur entwickelt, welche an die Charakteristika des Maxim Integrated Keyless Go System (KGO System) angepasst ist. Da diese Komponente den höchsten Qualitätsstandards der Automobilindustrie genügen muss, liegt ein weiterer Fokuspunkt der Arbeit auf der Verifikation der Prozedur sowie dem Test der Implementation.

Contents

Chapter 1

Introduction

Electronics in the automotive industry are getting more and more important, as the customer is demanding higher comfort and more functionality in cars. The range of application reaches from security features like Electronic Stability Program (ESP), usability features like automatically turning on the lights when unlocking, to entertainment systems within the car.

One big field is also the car locking/unlocking system, which has developed over time from basic mechanical keys, over remote keys to unlock or lock the car from distance, to the latest generation; which are the Keyless Go Systems, using smart keys known as Key-Fobs. Those Keyless Go Systems can automatically detect when to lock or unlock the car, without any user input. Therefore the user just has to carry the Intelligent Key Unit used in the Keyless Go System (Key-Fob), and does not need to take any action. The Maxim Integrated Keyless Go System (KGO System) focuses on the next generation of such a system, with enhanced functionality and improved security.

1.1 Problem

The Key-Fob is a mass product, whereas all Key-Fobs have the same initial state and just differ in the serial numbers identifying all components. The same assumption holds for the Base Station (BS) in the car. Therefore a procedure has to be found, that connects a specific Key-Fob with a specific car, respectively BS. To minimize the costs, this procedure should be done with a minimum of effort by an external handler.

1.2 Project Motivation

The automotive industry is a highly competitive field of economy, where keeping the product prices to a minimum is very important. Therefore, every component has to be chosen by the best quality to price ratio and also the assembling of those components must be easy. Saving money in assembling can mainly be done by automating the assembling procedures. This also involves the pairing of Key-Fobs with BS. At the moment car manufacturers use procedures that involve a high amount of manual effort to pair a Key-Fob to the BS. As the smart Key-Fobs also have some computational power (intelligence), this can be used to implement a procedure that automatically pairs a Key-Fob to a BS, minimizing the manual amount of effort for this task and therefore the costs.

1.3 Non-Functional Objectives

As part of an end product, which aims for a broad field of customers, there are several things that need to be taken into consideration when developing such systems, in order to get a high market impact:

- Ensure high usability and easy integration in the end product
- Low power consumption, as the end product does not have a permanent supply source and energy sustainability is an important issue nowadays.
- High Security and Quality Standards to ensure proper and reliable functionality of the product, as malfunctioning can cause big damage.
- In order to stay competitive, besides quality and functionality pricing is very important to the customer and the Original Equipment Manufacturer (OEM).

All those factors have been taken into consideration when developing the Maxim Integrated KGO System and influenced design decision, which is also observable within the context of the development of the Automatic Smart Key.

1.4 Functional Objectives and Resolution Method

The main objective of this Master Thesis is to find a procedure to automatically pairs Key-Fobs and BS tailored to the constraints of the KGO System. This is done by first analysing the KGO System and the pairing and collision management procedures, which are currently state-of-the-art. After the analysis a procedure is designed and implemented on the Hardware (HW) of the KGO System, whereas the focus also lies on testing and verifying the implementation.

1.5 Thesis Contents

Chapter 2 gives a short general insight to the development and advantages of the KGO System. In the Chapters 3 and 4, existing collision management and pairing procedures are analysed and reviewed whether they fit the Maxim Keyless Go System. The development of different approaches to solve the problem are explained and analysed closely in Chapter 5, in order to choose the better solution for the implementation. This theoretical definition is then followed by the chapters 6 and 7, which focus on the system constraints, implementation and testing of the implementation.

The final chapter 8 gives an overview of the development done in the context of this thesis, analysis them and shows development possibilities for future work.

Chapter 2

Maxim Integrated Keyless Go System

2.1 Functionality

The KGO System is the next generation of keyless entry and start systems for automotive applications. It offers a more secure wireless key solution, than current systems, including various additional functionalities.

2.1.1 Handling Multiple Key-Fobs

Due to different requirements of car-manufacturers and car rental agencies, this system is able to handle up to eight Key-Fobs per car. This implies that the system is able to detect and resolve collisions during communication in order to function properly.

2.1.2 Communication via UHF and LF or LF TRP

Due to the needs of the system regarding distance and usage two different Radio Frequency (RF) channels for communication, Low Frequency (LF) and Ultra High Frequency (UHF), are necessary. The main differences in terms of characteristics between LF and UHF communication are the energy consumption, which is higher for LF, the fact that LF travels through most of the obstacles, whereas UHF is blocked and the data rate which is higher for UHF transmissions. This characteristic also defines their use in the system; UHF is used for high distance communication between car and Key-Fob downlink[1] and

[1]The data-flow from BS to the Key-Fob is considered downlink.

uplink[2], for example to lock or unlock the car remotely via pressing a button on the Key-Fob. The LF channel, on the other hand, will be used for short range communication downlink and locating the Key-Fob. This is necessary to either verify if the Key-Fob is close enough to the car to unlock it or, in case of trying to start the car, if the Key-Fob is inside the car.

In addition, the LF Transponder (TRP) functionality also enables upstream communication via the LF channel on a very small range (around 10 cm) by using special antenna circuits on BS side and energy harvesting by the Key-Fob. This method is used to either save battery power or in case the Key-Fob unit lacks sufficient battery supply.

2.1.3 Door-handle Interface

To extend the functionality of the KGO System, the door-handles can be equipped with sensors. As those sensors may vary for each car-manufacturer, this system offers a universal intelligent door-handle interface, which increases flexible of the system.

2.1.4 Keyless Entry

As described above, the system is able to locate the Key-Fob relative to the car position. Therefore the car can be unlocked, if the door-handle is pulled and the Key-Fob is located close enough to the car to be sure no unauthorized person is trying to get into the car.

2.1.5 Keyless Start

Once the car-start button is pressed, the car determines if the Key-Fob is within the car and enables the start of the engine.

2.1.6 Self Diagnosis

Being able to ensure the functionality of all systems in an automotive environment are very critical. Therefore all components of the KGO System include self diagnosis interfaces, to verify their proper functionality during operation without any further effort by an operator.

[2]The data-flow from Key-Fob to the BS is considered uplink.

2.2 Architecture

2.2.1 System Overview

The system consists of a BS, located in the car body, and a Key-Fob unit. The BS is responsible for managing all car sensors (e.g. door-handle) and communication with the car electronics as well as with the Key-Fob unit. To exchange data from BS to the Key-Fob unit, wireless communication via UHF or LF channel is used.

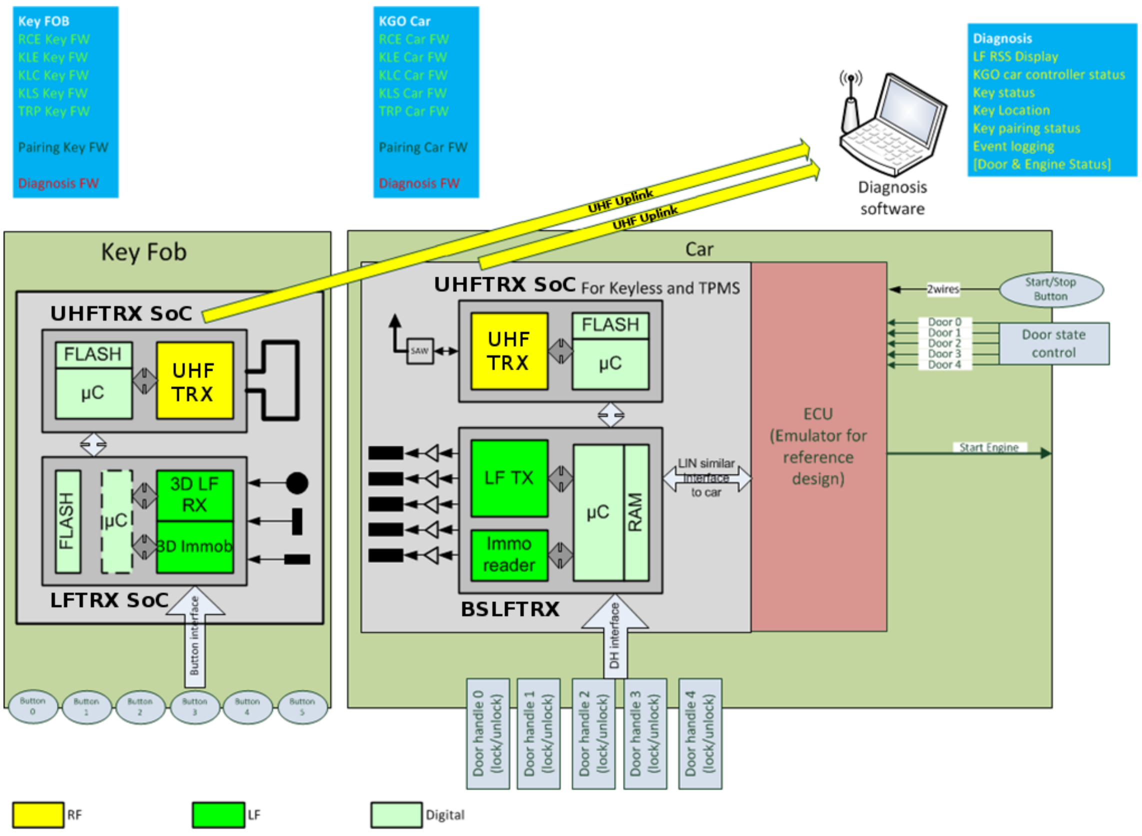

Figure 2.1: This block diagram illustrates the components of the KGO System in a reference design.

2.2.2 In-Car Construction

A part of the system is included in the car body and consists of four LF antennas and one LF TRP antenna[3], which are located as illustrated in figure 2.2. The four LF-antennas are

[3]The LF TRP antenna consists of the same coil as a standard LF antenna, but the capacitor is not included in the housing. The capacitor needed in the sending circuit of the LF TRP antenna can be

necessary to cover the whole area around the car with sufficient LF field for positioning, therefore their position is fixed. The position of the TRP antenna, on the other hand, can vary for each car manufacturer, figure 2.2 just shows a reference design build by Maxim Integrated.

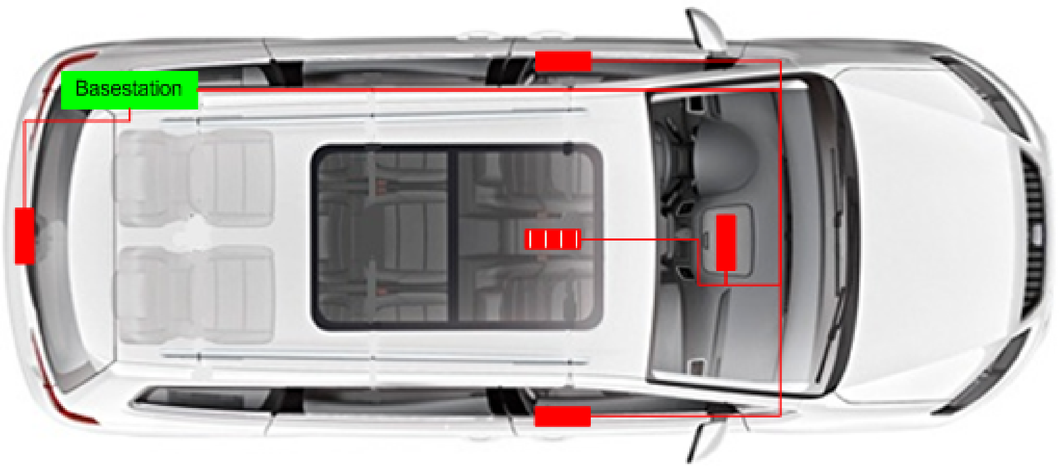

Figure 2.2: Position of the LF antennas within the car, the patterned square represents a possible position for the LF TRP antenna [SEA14].

2.3 Analysis of Single System Components

2.3.1 Base Station LF Transceiver SoC (BSLFTRX)

The BSLFTRX is the core of the KGO System. It manages the door-handle operations, the LF TRP- and LF communication as well as the communication to the car-control system and the UHF Transceiver (TRX), which is also part of the BS. Therefore this System-on-Chip (SoC) implements following interfaces:

- An intelligent Door-handle Interface
- LF-Antennas Interface
- 3-Wire-Interface (3WI)) or Universal Asynchronous Receiver/Transmitter (UART) (implemented as Single Wire Interface (SWI))
- General Purpose Input Output (GPIO) Pins
- Various Control and Measurement Interfaces

connected or disconnected in the BS.

- Joint Test Action Group (JTAG) Debug Interface
- Central Processing Unit (CPU) Interfaces like AMBA Peripheral Bus (APB) or JTAG Like Control Chain (JLCC)
- Several Special Function Pins.

The Firmware (FW) of the chip runs on an integrated microcontroller which has access to Read-Only Memory (ROM) and Random Access Memory (RAM) memory. Also dedicated Digital Signal Processing (DSP) parts and sate-machines, handling specific tasks, are included in the SoC, in order to reduce the CPU load.

LF Data Transmission

The LF data transmission can either be controlled by the CPU as well as a state machine, whereas the manual mode has the advantage that the packet length is not limited to buffer length[4]. The antenna drivers, in order to create the signal carrier, and the modulation output for Binary Phase Shift Keying (BPSK) and Quadrature Shift Keying (QPSK) are always controlled by a state machine, which runs parallel to the CPU. The LF BS is able to transmit and receive data via the LF link, whereas for transmitting the quality of the LF antenna circuit has to be changed. The BSLFTRX supports Amplitude Shift Keying (ASK) as well as BPSK and QPSK with differential encoding.

Current Control Unit and Immobilizer

The Current Control Unit and Immobilizer of the SoC are used to measure the current through the LF antenna, to be able to control the sending power. This unit includes an Analog to Digital Converter (ADC) as well as peripheral blocks to enable the following functionalities:

- Offset compensation to keep high resolution for small signals.
- Sufficient oversampling at small signals.
- Measuring non-sinusoidal signals.
- Direct Current (DC) as well as LF signal magnitude and phase measurements
- Temperature sensors

[4]If the buffer of the state-machine is reloaded, the LF datastream is discontinued. In manual mode data can be reloaded without discontinuing the LF datastream

Door-handle Interface

The door-handle interface can connect several doorhandle types to the SoC. It is possible to supply the doorhandle with power as well as a single sense line to get information from the doorhandle to the CPU. Also a simple communication protocol, used by the doorhandle, can be implemented in the FW of the CPU.

Oscillators

Due to power management and clock distribution reasons, the SoC has several different clock sources. Namely: a watchdog oscillator, a RC Oscillator (RCO) and a crystal oscillator.

Diagnostics

In order to diagnose the door-handles and antennas, sufficient measurement interfaces are implemented. This is necessary to be able to check the proper functionality of these peripherals without any additional HW setup.

Self-Test, Debug and Self-Protection

Build-In-Self-Test (BIST), RAM parity test, test pins, overheat protection and a JTAG debug interface are also included in the SoC, in order to increase usability and guarantee functionality.

2.3.2 Key-Fob LF Transceiver SoC (LFTRX)

The LFTRX, used in the Key-Fob units, is a ultra low power SoC designed for long term mobile use. The SoC includes following interfaces and components:

- 3D LF Interface
- 3WI
- Event Interface
- Measurement Interface
- External Supply Interface
- Transparent Data Interface

- Debugging Interface
- Parallel System Bus (internal)
- 5-Button Interface for User Interaction
- Advanced Encryption Standard (AES) Cryptographic Engine
- Unique 32-bit Identification number (ID)

3D LF Interface

The SoC is a 3D LF start-up receiver and TRP. This means the SoC can be woken via a LF or LF TRP start-up signal and the relative position of the Key-Fob to the BS LF antenna does not influence the receiving quality, as receiving is possible in X, Y and Z channel (respectively direction). Furthermore the SoC automatically finds the best channel for LF or LF TRP communication, according to the signal strength. The TRX and TRP supports (D)BPSK or ASK with PPM20 or Manchester coding using active modulation.

The LF TRP communication is used in case the battery supply is insufficient or to extend battery life. LF TRX and TRP use power monitoring and power management, to work energy efficient and harvest enough energy from the LF coil to generate a supply voltage.

Event Interface

This interface gathers various external events and routes them internally to the CPU. This can be used for e.g. to trigger power management or the start-up.

Measurement Interface

Consisting of an Analog IO Pin for Received Signal Strength Indication (RSSI), an ADC, a VDD measurement unit and two temperature sensors. The measurement interface has the purpose of supplying the system with sufficient environmental data to perform temperature and VDD matching, to increase RSSI measurement accuracy.

Transparent Data and Debugging Interface

For developing purpose, the SoC also includes various GPIO pins and control signals, as well as a JTAG debug interface.

Oscillators

The only clock source available in this SoC is a RCO with a maximum drift of ±2% from the nominal frequency.

Power Management

Due to the desire of low power operation of this SoC, it implements several supply units, which can be switched on and off, regarding the current usage of the SoC, to keep the power consumption to a minimum.

2.3.3 UHF Transceiver SoC (UHFTRX)

The UHF TRX SoC consists of a microcontroller as well as several interfaces and peripheral blocks or state machines to fulfill specific tasks and decrease the CPU load. For instance:

- 3WI or UART
- Mixed Signal Sensor Interface for up to 10 freely configurable Analog/Digital Input/Output (I/O) Pins
- GPIO Pins
- CPU interfaces like APB or JLCC
- JTAG Debug Interface
- Random Number Generator for Security Applications
- Temperature Sensors

There are asynchronous transmission protocol for ASK and pleioynchronous protocol for Frequency Shift Keying (FSK) (or Minimum Shift Keying (MSK)) supported by the SoC in the quad bands of:

- 315 MHz
- 433 MHz
- 868 MHz
- 915 MHz

Furthermore the SoC can work as master device controlling peripherals or as slave being controlled by more powerful SoC or microcontroller and also implements several sleeping modes for power saving in different applications. This UHF TRX SoC is used in the BS as well as the Key-Fob of the KGO System.

2.3.4 Usage of the Components within the Context of this Project

This project focuses on the LF TRP communication, therefore mainly the BSLFTRX and the LFTRX will be used. The UHF TRX will just be used for programming the BSLFTRX, as for development purposes the FW of the BSLFTRX is stored in RAM.

Chapter 3

Analysis of State-of-the-Art Collision Management Algorithms

As in most wireless systems, also the Maxim Integrated KGO System offers the possibility to use multiple clients (Key-Fobs) with one host (BS). Therefore multiple access methods and collision management are needed to ensure proper communication between all clients and the host system. In this chapter a brief overview over multiple access methods is given and some collision management strategies, used in wireless communication,n will be discussed. Also a theoretical overview on those methods and strategies will be given. The focus, although, lies on processes already used in similar wireless systems like Radio Frequency Identification (RFID) [Fin02].

3.1 Multiple Access

The goal of multiple access methods is to divide one dimension of a communication channel in a way, that it is possible for multiple clients to use this channel together. A communication channel can be divided into four dimensions and therefore there are four basic multiple access methods, as explained in [Fin02]:

- Space → Space Division Multiple Access (SDMA)
- Time → Time Division Multiple Access (TDMA)
- Frequency → Frequency Division Multiple Access (FDMA)
- Code → Code Division Multiple Access (CDMA)

Basically every method slices a dimension of the channel into slots and then assigns one slot to a client, so this client can use the channel in/during this slot.

3.1.1 SDMA

The space around the host is divided into segments (for e.g. using directional antennas) whereas one segment can be occupied by one client at a time. A constant data stream is possible as long as the client stays in the same space-segment.

3.1.2 TDMA

A sequential series of timeslots are defined, with specific length and starting times. In each timeslot, one client can communicate with the host. Constant data streaming is not possible as soon as the timeslots are shorter than the data-packets.

3.1.3 FDMA

The host offers multiple frequency channels which can be used by the clients at the time, which allows a constant data-stream.

3.1.4 CDMA

This mathematical method involves the usage of orthogonal coding, so the channel is split in the fourth dimension - the code dimension. The data sent by a client to the host is coded using an orthogonal spreading code. Due to this special coding the host is able to differentiate between different data, send by different clients. Using the same physical channel (space, time and frequency dimension do not differ), by decoding the data also using the same system of orthogonal spreading codes. This multiple access method also allows a constant data stream.

3.2 Anti-Collision Processes

Even when using multiple access methods, there is the need for anti-collision handling. This is necessary, if the assignment of communication slots is not done properly or there are more clients than communication slots, so multiple clients try to communicate within one communication slot.

3.2.1 ALOHA

The ALOHA[1] process uses TDMA to enable multiple access to a communication channel. Each client starts its communication at a random time, for instance when the data-packet is ready. The performance of this stochastic process is highly dependent on the number of clients and packets sent during a specified period of time. In [Fin02] the performance was rated as specified in Eq. 3.1 to Eq. 3.3.

$N \cdots$ number of clients

$T \cdots$ duration of the observation period

$\tau_n \cdots$ duration of a packet from client n

$r_n \cdots$ number of packets sent from client n

$G \cdots$ traffic

$S \cdots$ channel capacity

$q \cdots$ success rate - probability of a packet to be send without collision

$$G = \sum \frac{\tau_n}{T} \cdot r_n{}_{n=1}^{N} \tag{3.1}$$

$$S = G \cdot e^{-2 \cdot G} \tag{3.2}$$

$$q = \frac{S}{G} = e^{-2 \cdot G} \tag{3.3}$$

Figure 3.1 illustrates the capacity of the channel versus the traffic using ALOHA. It can be seen that the maximum capacity of 18.4% is reached with a traffic G of 0.5. This means that the time used to send packages equals $\frac{T}{2}$, so the channel is just used half of the time.

3.2.2 Slotted ALOHA

The Slotted ALOHA (SALOHA) procedure works similar to the ALOHA procedure, with the difference that the time period T is now divided into timeslots. Therefore the clients cannot send their data at any random time, they are just allowed to start sending data at the beginning of a timeslot. A time synchronization for all clients and the host is necessary for this procedure. This increases the capacity and the maximum traffic which can be seen in figure 3.1.

As remarked in [Fin02], due to the timeslots, the collision interval T_c is halved. Considering equally long packets of length τ, using the ALOHA procedure, a collision can occur if the

[1]The name origins from the fact that this process was originally developed for the ALOHANET in the 1970ies.

sending interval of two clients T_s is smaller than $2 \cdot \tau$. Therefore the collision interval T_c equals $2 \cdot \tau$. Using SALOHA, a collision can just occur within a interval of τ, as sending of a packet has to start at the beginning of a timeslot. This means that the T_c equals τ for SALOHA and moreover the capacity of the SALOHA system [Fin02] is

$$S = G \cdot e^{-G} \tag{3.4}$$

which maximum is 36.8% at a traffic of $G = 1$.

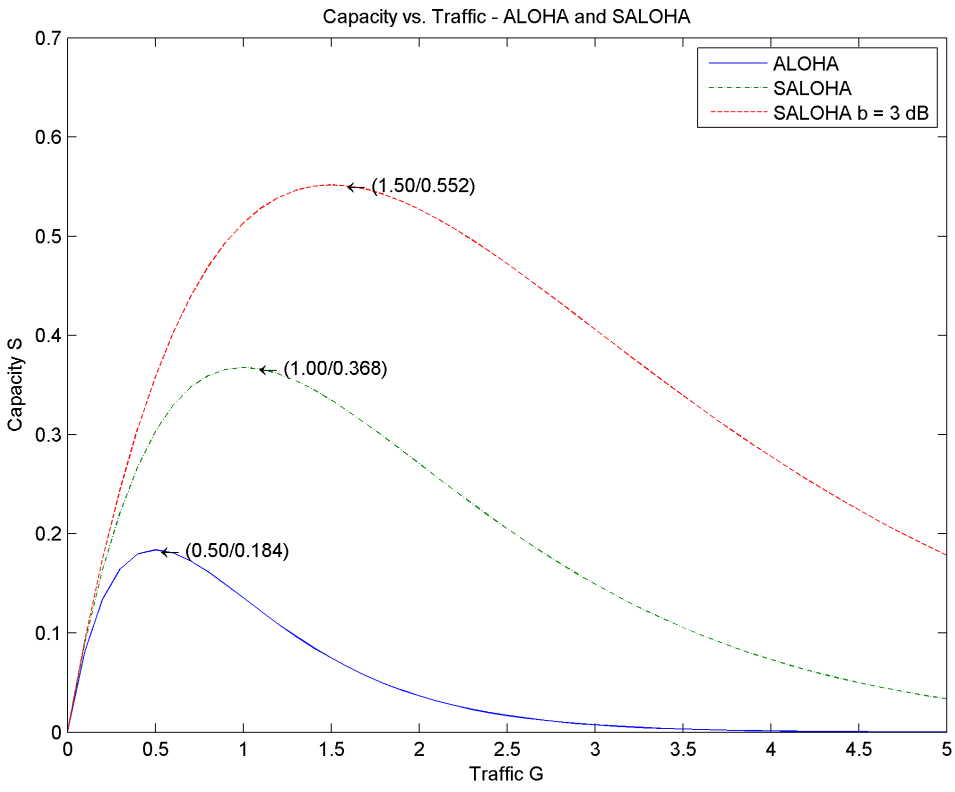

Figure 3.1: Channel capacity versus traffic using a ALOHA based collision management. The capacity decreases drastically as soon as the maximum is reached.

Capture Effect

In addition, the capture effect needs to be considered. In wireless systems it can happen that collisions are not recognized because one sender simply “drowns out” the other ones. This can improve the capacity of the channel as there are fewer collisions (data transmitted is valid), but can also cause the “weak” clients to not be able to send their data. The parameter for this effect is, concerning [Fin02], the threshold b. It specifies how much stronger a packet has to be than another one, in order to be detected without any error.

$$S = G \cdot e^{-\frac{b \cdot G}{1+B}} \tag{3.5}$$

The effect is illustrated in figure 3.1, whereas a threshold b of 3 dB was chosen.

Dynamic SALOHA

As explained at the beginning of section 3.2.2, the SALOHA procedure has the best performance for a traffic G of 1. This means that the number of timeslots equals the number of clients, but in practice most of the time the exact number of clients cannot be determined in advance. If there are more clients than timeslots, the capacity converges towards zero very fast, but foreseeing many timeslots for just a few clients is also not good in terms of performance. Therefore in practice several procedures where implemented to offer a variable number of timeslots to the clients. For more detailed information see [Fin02] *chapter 7.2.4.2.1.*

In practice there are also many different variations of the ALOHA protocol used, to work with multichannel systems [SL03], improved by using advanced technologies like spread spectrum coding [KH99] or Direct Sequence-CDMA (DS-CDMA) [LZ94].

3.2.3 Binary Search Based

The binary search based collision management procedure requires the implementation of the binary search algorithm as well as proper encoding of the data and the ability to detect the bit-position when the collision occurs. Proper encoding could be for e.g. No-Return-to-Zero (NRZ) or Manchester [Fin02]. The algorithm itself consists of a flow of requests from the host and replies by the clients with the goal of selecting one client from a group. With a request, the host can select one or multiple clients depending on the requested address or ID.

To illustrate the function of a binary search based procedure, all possible client addresses or IDs are represented within a ID-tree (see figure 3.2). In each iteration it can be determined on which bit-position of the address a bit error occurred. Therefore we can find a path from the top of the ID-tree to the bottom. Table 3.1 and 3.2 show the iteration steps for binary search scenario.

	Address
TRP 1	1001
TRP 2	1110
TRP 3	1011
TRP 4	1101

Table 3.1: IDs of the transponders used in the binary search example.

Iteration	Request Address	Replying Transponders	Received IDs
1	$\leq$ 1111	1, 2, 3, 4	1XXX
2	$\leq$ 1011	1, 3	10X1
3	$\leq$ 1001	1	1001

Table 3.2: Development of the received IDs in each iteration. X indicates a collision.

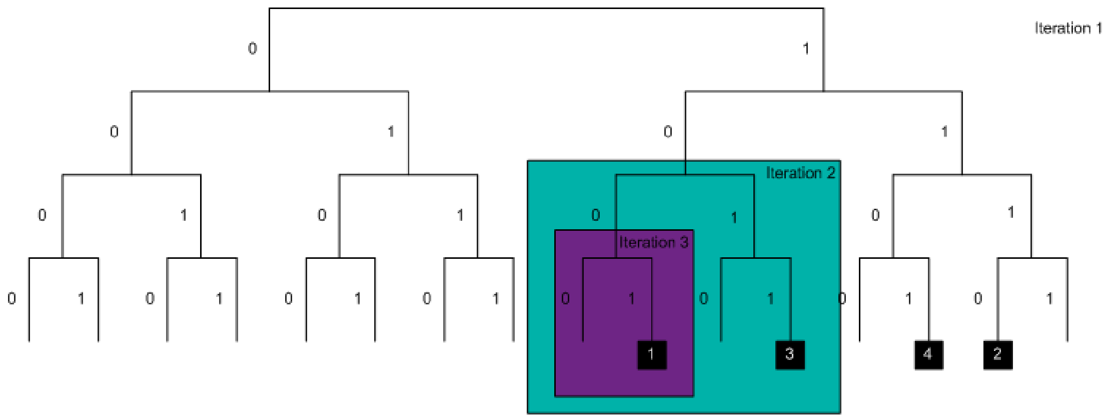

Figure 3.2: Example for a ID Search tree.

In practice for every iteration, not the whole ID is transmitted, just a small portion of it. Therefore the amount of data need to be send between the host and the clients decreases as just parts of the ID are transmitted. In average, a binary search algorithm like its illustrated here, needs

$$L(N) = ld(N) + 1 = \frac{log(N)}{log(2)} + 1 \tag{3.6}$$

iterations [Fin02], whereas N represents the number of clients.

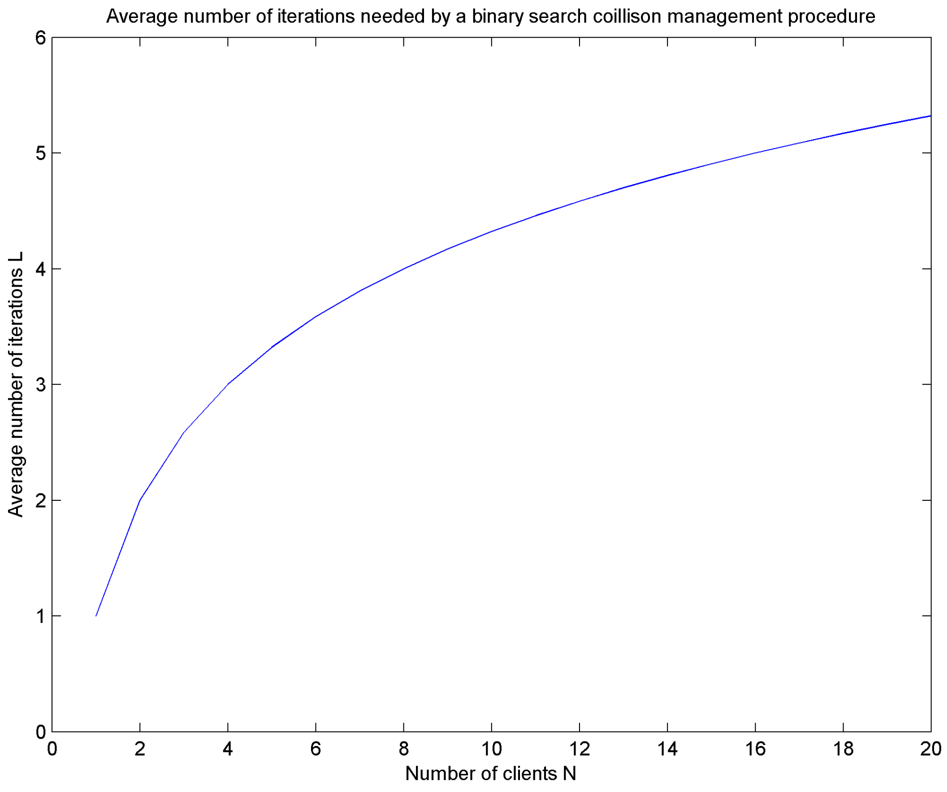

Figure 3.3: Average number of iterations needed vs. number of clients for a binary search collision management process.

3.2.4 Other

CSMA/CD

This Collision Sense Multiple Access with Collision Detection (CSMA/CD) procedure is mainly used in wired Ethernet standards. It relies on fast networks and randomized access control with no arbiter. Whenever a packet is ready, the client senses the channel and starts sending if the channel is free. In case of a collision or an occupied channel, the client delays the restart for a specific amount of time.

CSMA/CA

Collision Sense Multiple Access with Collision Avoidance (CSMA/CA) is used in most of the WLAN standards. Similar to CSMA/CD the client first senses if the channel is free before he starts sending. Contrary to CSMA/CD now the client does not send the data immediately, it sends a Ready-to-Send (RTS) packet to the desired host. Transmission can just be started if the client receives a Clear-to-Send (CTS) as reply to his RTS from the host. The whole data transmission then has to be acknowledged by the host to ensure collision-free transmission, this is necessary as its hardly possible for clients to listen to a wireless channel while they are sending.

Chapter 4

Analysis of State-of-the-Art Pairing Procedures

Pairing defines a process which links two devices together to communicate over a secure channel. While the usage of ubiquitous systems like Bluetooth, ZigBee or WiFi increases more and more, also the diversity of different approaches on secure pairing methods increases [KSTU09]. With this high diversity of methods, there comes one major drawback: there is no global security infrastructure existing yet. Such an infrastructure would be necessary, as the devices do not share any common knowledge with each other before the communication setup, in contrast to many other communication systems. Many pairing procedures therefore use a Out-Of-Bound Channel (OOB), which mostly involve user interaction. A OOB can be physical, visual, auditive or sensitive which can be vulnerable to Man-In-The-Middle and Evil-Twin attacks. Also the usage of a OOB forces the developers to keep usability to a maximum, as this minimizes the probability for human errors, which most times leads to a decrease in security. Sometimes little things, like using hyphens to separate PIN letters [KS09] can increase robustness and usability.

This chapter mainly focuses on explaining the pairing algorithms used in Bluetooth as those algorithms involve a minimum of user interaction.

4.1 Link Manager Protocol

The Link Manager Protocol (LMP) pairing is initialized by device A (Verifier/Initiator) after user input, by generating a random number and sending it to device B (Claimant) [IEE02]. Device B acknowledges the random number with the LMP_accepted packet. Now both users have to type the identical PIN (in case of Bluetooth 4 digits) into device A and B, which is used to calculate the link key K_{init}. If the PINs do not match the link key will also not match and the pairing will fail. In case the link keys K_{init} match, the LMP-authentication will succeed.

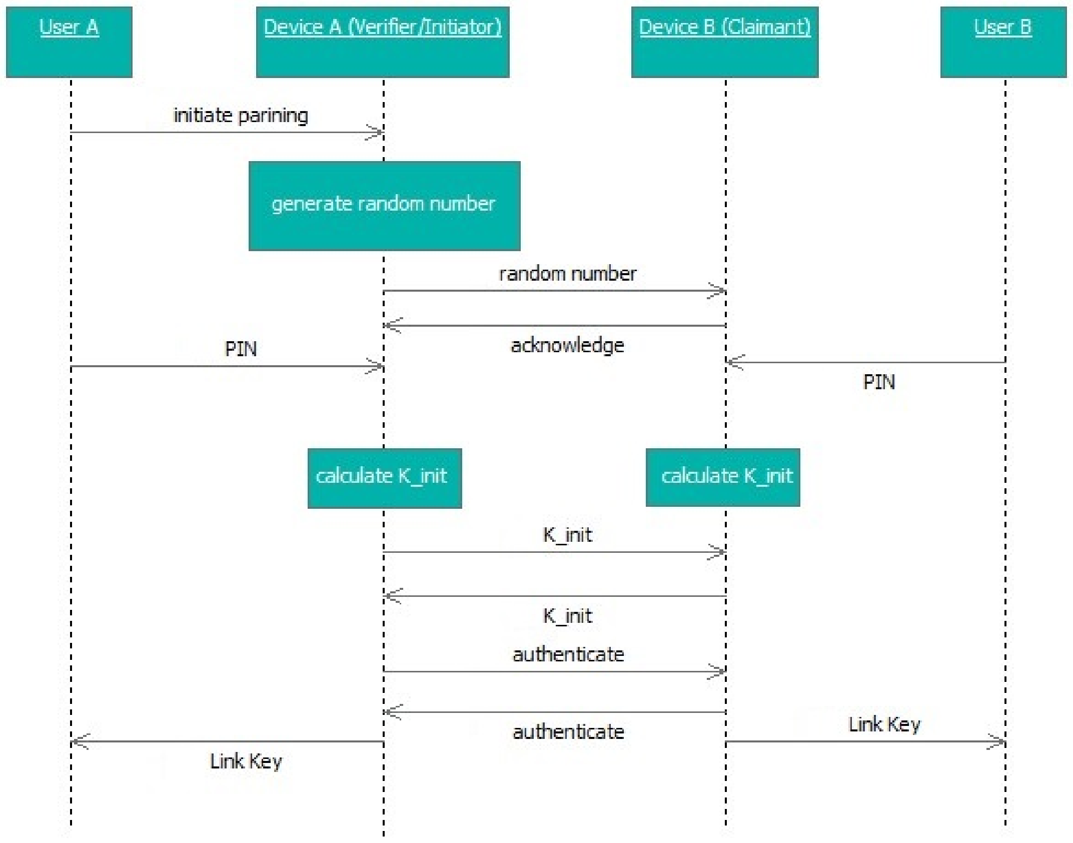

Figure 4.1: Sequence diagram of the Bluetooth LMP-Pairing.

It needs to be mentioned that the K_{init} is not used for further communication. The devices use K_{init} just to exchange the necessary credentials to build a secure channel, for further communication a link key is generated.

4.2 Secure Simple Pairing

Secure Simple Pairing (SSP) is the second generation pairing algorithm used in Bluetooth to increase security during the pairing process. This procedure uses Elliptic Curve Cryptography (ECC) and therefore there is no need for a PIN input anymore, whereas its not prohibited to use an additional PIN input. Furthermore the value range of the link key

K_{init} is extended over 2^{128} possibilities [Ell11], which makes it even harder to crack.

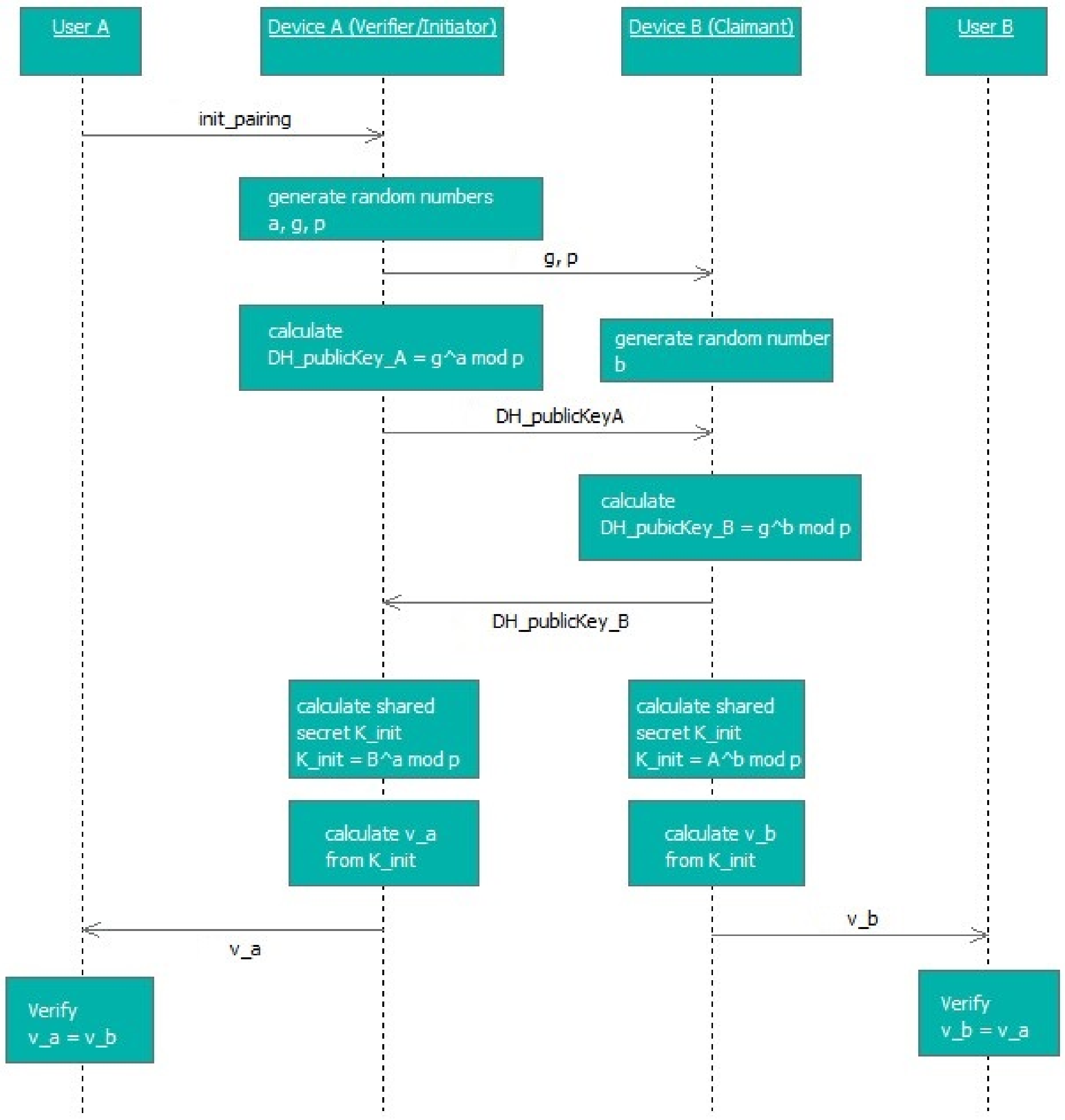
User A
Device A (Verifier/Initiator)
Device B (Claimant)
User B
init_pairing
generate random numbers a, g, p
g, p
calculate DH_publicKey_A = g^a mod p
generate random number b
DH_publicKeyA
calculate DH_pubicKey_B = g^b mod p
DH_publicKey_B
calculate shared secret K_init K_init = B^a mod p
calculate shared secret K_init K_init = A^b mod p
calculate v_a from K_init
calculate v_b from K_init
v_b
v_a
Verify v_a = v_b
Verify v_b = v_a

Figure 4.2: Sequence diagram of the Bluetooth SSP.

At the beginning of SSP, the two devices establish a shared secret channel using the Diffie-Hellman procedure, then sharing the Diffie-Hellman Key which is a 192 bit random number [Ell11]. The random numbers are generated using elliptic curves from ECC. Each device then uses the shared secret K_{init} to calculate the control value v_a or rather v_b, which is displayed to the user and needs to be confirmed to finalize the pairing process.

4.3 Other Aspects

As mentioned before, mostly OOBs are used in pairing procedures to ensure security. Those OOB can sometimes involve special HW like buttons, cameras, microphones or displays [SEKA11] which are not available in all systems. Therefore not every pairing procedure can be used in every systems. Some systems also use physically secure communication channels like cables to initialize pairing.

For devices with higher computational power and displays, also games could be used for pairing. In [GSV11] pairing based on the memory game “Simon says” is proposed, to decrease the probability of human errors and maximize the security. Humans will be more focused on a game, and therefore become less prone to make mistakes, than just on a simple number input like in other pairing procedures.

Using all those OOB, also attacks on those, mostly uncontrolled, channels need to be taken into consideration. Especially when using auditive channels, as the attacker can be easily invisible for the user [HS13]. Therefore it needs a lot of investigation to make the OOB channel secure.

Chapter 5

Development of a Pairing Procedure with Collision Management for multiple Key-Fobs

Due to the special characteristics of the KGO System a pairing procedure or collision management procedure from literature or other technologies cannot be applied without modification. Therefore, this chapter describes the development of ay specific pairing procedure with collision management for KGO System. For this theoretical definition of the procedure a working HW setup is assumed, and therefore not all HW failure cases are described and modeled here. This is done in order to keep the focus on the procedure itself and not the error handling. HW errors will be discussed chapter 7.

5.1 Analysis of Pairing and Collision Management Strategies

Considering the already known collision management strategies, introduce in chapter 3, certain strategies can already be excluded. First, CSMA/CD and CSMA/CA are not fitting to the application due to the fact that neither Key-Fobs nor the BS can perform Transmit (TX) and Receive (RX) at the same time, so they are not able to sense collisions while they are transmitting. This is also mentioned as one of the main criteria for the section of collision management procedures in [LG08]. Furthermore a pure Binary Search based algorithm can also not be implemented, as in the KGO System, a bit-wise error detection is not possible. This is due to the lack of a global time-source, which makes it very unlikely that colliding packets are aligned in time.

The pairing includes the collision management and is completed when the credentials between car (BS) and Key-Fob are exchanged, which consist of the car ID and the cryptographic key. When and how often the pairing procedure is triggered will not be specified in this thesis, as this is up to the OEM and should be defined by them. The only restriction is, that the procedure can only be triggered if the Key-Fobs are in LF TRP communication range of the BS. This is also the main security feature, as for this thesis it is assumed that the pairing procedure can just be done within a secure environment, for example the factory. Therefore the range of the LF TRP will always lie within the factory and cannot be interfered by any attacker. For this reason no further security codes, encryption or other security feature are implemented now.

The following sections will describe the procedure in detail.

5.2 Collision Identification

The basic need for a collision management system is the identifications of collisions. In the KGO System just error detection, but no error correction is needed. Moreover, due to the low computational power of the LFTRX SoC used on the Key-Fob and the timing constrains, no complex solutions can be chosen. Considering those constrains and after evaluating the results of different checksums and error detection codes in [Ngu05] and [MK09] the Fletcher-Checksum has been identified as the best fitting solution for the KGO System.

The Fletcher-Checksum offers a simple calculation method, which can easily be done with a low computational effort. Furthermore it has a low probability of undetected errors for short code word lengths, compared with a simple Cycle Redundancy Check (CRC) checksum [MK09].

5.3 Timeslot Definition

The KGO System is designed to use only broadcast messages for BS to Key-Fob LF TRP communication and TDMA for Key-Fob to BS LF TRP communication. Due to the lack of a global time source and the inaccuracy of the RCO, evaluated in the Laboratory, ($\Delta f_{RCO} = \pm 2\% \cdot f_{RCO}$) the timeslots cannot be aligned equally. Considering the lowest possible clocked Key-Fob ($f_{RCO-actual} = f_{RCO} - \Delta f_{RCO}$), the optimal clocked Key-Fob ($f_{RCO-actual} = f_{RCO}$) and the fastest possible clocked Key-Fob ($f_{RCO-actual} = f_{RCO} +$

Δf_{RCO}) as well as the inaccuracies of the timers in the Key-Fobs and the BS, the timeslots shown in figure 5.1 and in table 5.1 have been calculated. For this calculations a ideal guard time of 2.87 ms has been chosen, due to the timer resolution in the Key-Fob and the BS. Of course, as all timings, also this guard time changes if Δf_{RCO} is not zero. Therefore the real guard time between the timeslots is less, as table 5.1 contains the worst case starting times (fastest possible Key-Fob) and stop times (slowest possible Key-Fob).

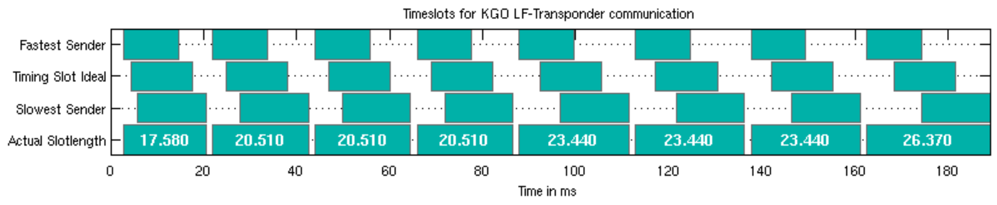

Figure 5.1: Timeslots used in KGO System.

Timeslot	Start Time	Stop Time	BS Start Timer-Value	BS Stop Timer-Value
	ms	ms	ticks	ticks
1	2.93	20.5	2	14
2	22	42.5	15	29
3	44	64.5	30	44
4	65.9	86.4	45	59
5	87.9	111	60	76
6	113	136	77	93
7	138	161	94	110
8	163	189	111	129

Table 5.1: Actual timeslot timings used in the KGO System.

5.4 Pairing Procedure with Stochastic Collision Management

The first approach is to use a pure ALOHA procedure with random number sources for LF TRP communication timeslot selection by the Key-Fob.

5.4.1 Formal Algorithm Definition

Before the timeslot of the Key-Fob is fixed by the BS, the Key-Fob uses a random number source to determine the reply-slot-number. This could be done by doing a ADC measurement on VTP or VDD. This measurement is done for every reply, therefore every Key-Fob re-evaluates the reply timeslot for every packet.

5.4.2 Base Station Functionality

The BS controls the procedure and takes all the decisions. For that, the BS holds a command-set described in table 5.2.

Base Station Commands

Command	Parameter	Descriptions
REQUEST	void	Requests all Key-Fobs.
SET_TIMING	ID timeslot number	Sets the timeslot number and SEND_LOCK for a certain Key-Fob specified by the ID.
START_CRED_EXHANGE	ID	Start credential exchange between BS and Key-Fob. Address 0 is a broadcast.
CLEAR	void	Pairing failed - reset all configuration registers to enable a reuse of the Key-Fobs.

Table 5.2: Description of the commands used by the BS.

Base Station Datastructures

Data Structure	Descriptions
Timing Table	Maps the IDs of the successfully identified Key-Fobs to the later on used timeslots.
Reply Table	In this table the replies of the Key-Fobs to a REQUEST command are saved.

Table 5.3: Datastructures held by the BS.

Timing Table The Timing Table is an $8x3$ table, holding triple of (Timeslot number, Key-Fob ID, valid flag). This table is used to track which Key-Fob will be assigned to which timeslot in the SET TIMING step, of the pairing procedure. Inserting, deleting and reading elements from this table are normal array operations.

Timslot Number	Key-Fob ID	Valid Flag
Final timeslot number assigned to the Key-Fob with SET_TIMING.	Full ID of the Key-Fob, that is assigned to this timeslot.	Indicates if entry is valid.

Table 5.4: Timing Table structure.

Reply Table The Reply Table is an $8x3$ table, holding sets of (Timeslot number, reply Key-Fob ID, Checksum). This table is used to track which Key-Fob replied in which timeslot to a REQUEST package. Inserting, deleting and reading elements from this table are normal array operations.

Timslot Number	Reply Key-Fob ID	Checksum
Number of the timeslot, the reply was received in.	Key-Fob ID received in this timeslot.	Checksum received in the packet with the Key-Fob ID to proof the correctness of the packet.

Table 5.5: Reply Table structure.

Execution Flow of the Base Station

5.4.3 Key-Fob Functionality

The Key-Fob is event driven by the commands received from the BS. The events can be seen in table 5.6.

Key-Fob Event command received	Key-Fob Action
REQUEST	IF SEND_LOCK not set: find reply timeslot, reply with key-ID package.
SET_TIMING	IF ID check passed: set timeslot number and SEND_LOCK.
START_CRED_EXHANGE	Start credential exchange.
CLEAR	Reset all configuration registers (timeslot number, etc.).

Table 5.6: Description of the Key-Fob events triggered by received commands.

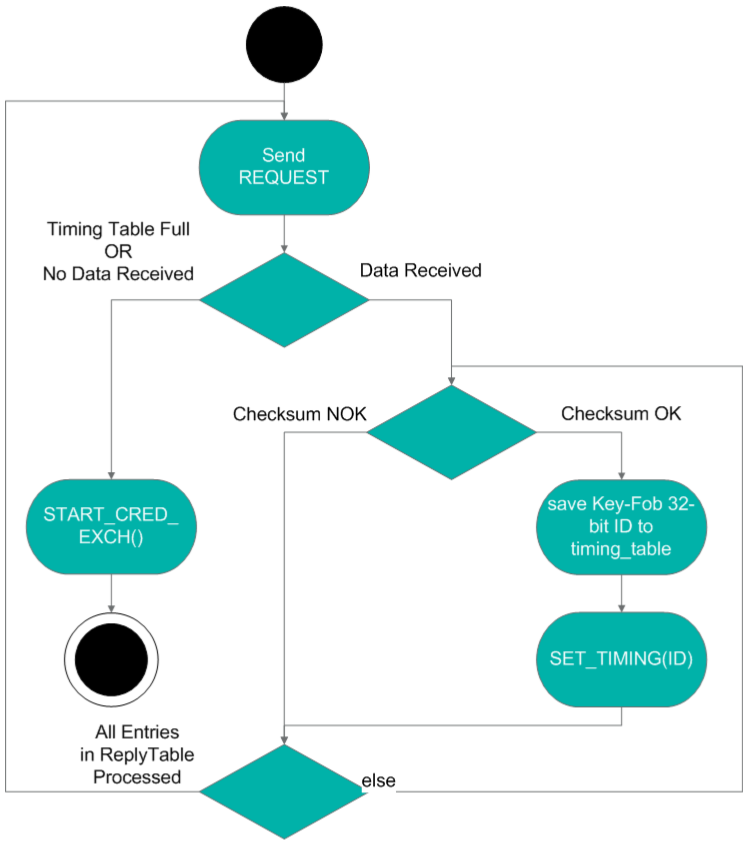
Send REQUEST
Timing Table Full OR No Data Received
Data Received
Checksum NOK
Checksum OK
START_CRED_EXCH()
save Key-Fob 32-bit ID to timing_table
SET_TIMING(ID)
All Entries in ReplyTable Processed
else

Figure 5.2: Execution flow in the BS, of the stochastic collision management procedure.

5.4.4 Example

Consider a system with three Key-Fobs and one BS, and the communication sequence as illustrated in figure 5.3. The BS wants to pair with all Key-Fobs, therefore it sends a REQUEST. All Key-Fobs receive the REQUEST and do a measurement, to determine the reply-timeslot. All three Key-Fobs reply, whereas the Key-Fob B and Key-Fob C choose the same reply timeslot. Therefore, when the BS processes the reply table, it recognizes a collision and successfully detect Key-Fob A. Now Key-Fob A is registered in the timing table and a SET_TIMING command is sent to the Key-Fob, so the permanent timeslot and the SEND_LOCK of the Key-Fob are set. The SEND_LOCK is used to prevent the Key-Fob from replying to further REQUEST commands and therefore cause further collisions.

Now again a REQUEST broadcast is sent, this time just Key-Fob B and C reply and choose different timeslots, so they are successfully detected. The timing table is again updated and the SET_TIMING command sent to both Key-Fobs B and C. Before starting the credential exchange, the BS again sends a REQUEST broadcast to check if there are

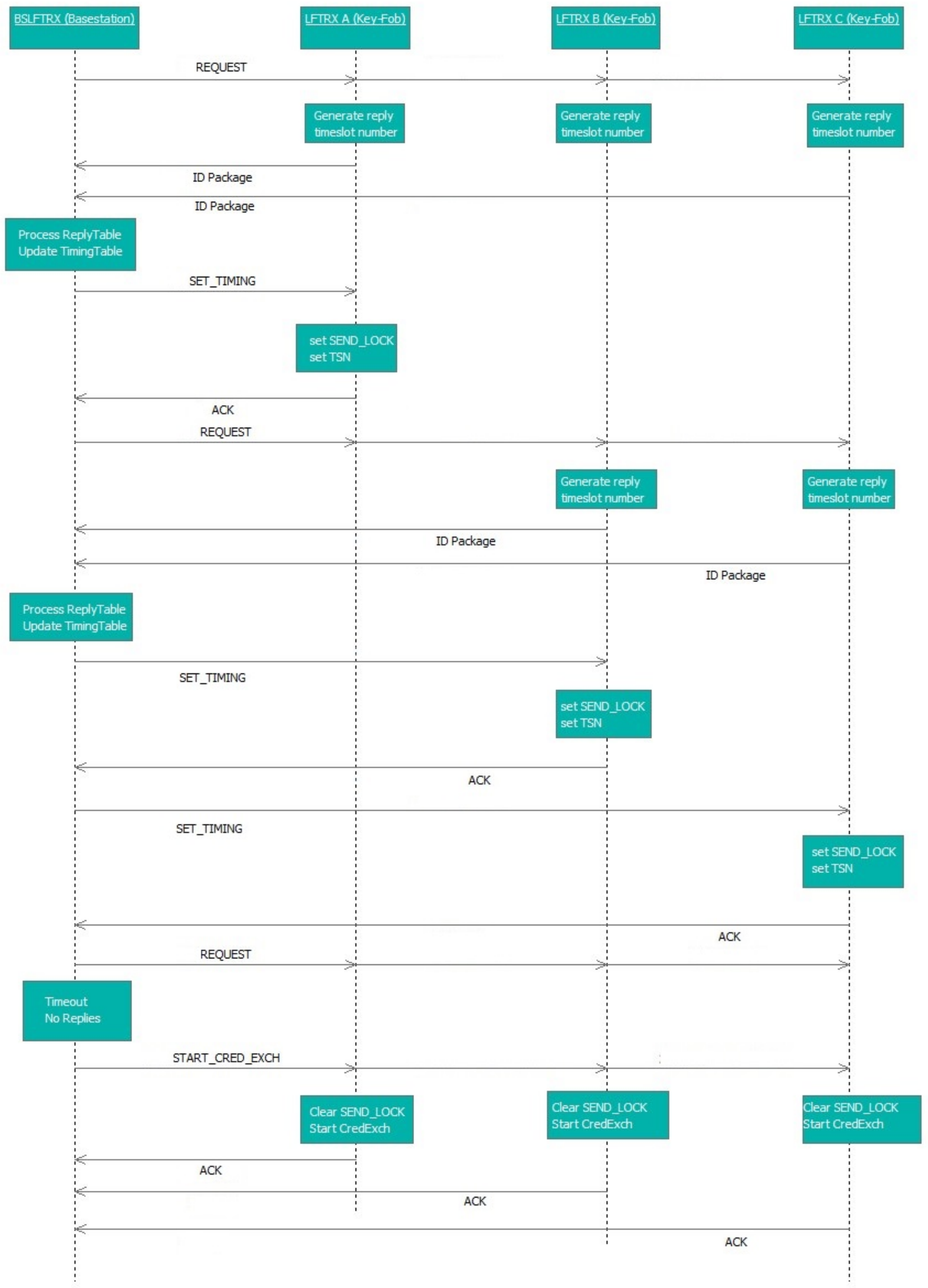

Figure 5.3: Communication sequence for the example of the pairing procedure with stochastic collision management.

any Key-Fobs in range, which have not been registered yet. If not (timeout occurs), the credential exchange is started and the process is finished.

5.4.5 Proof of Correctness

If the random number sources are sufficient the procedure is correct, as there will be a distinction of all Key-Fobs at one point, due to randomness of the number source which chooses the reply timeslot.

5.4.6 Proof of Termination

If the random number sources are sufficient the procedure will terminate, as there will be a distinction of all Key-Fobs at one point, due to randomness of the number source. A deadlock cannot occur.

5.4.7 Run Time Analysis

The duration of this procedure depends on the random number source (most likely an ADC measurement and possible calculations done with the ADC value afterward). Therefore, a run time analysis cannot be done, until the random number source, which is used, is identified.

5.4.8 Requirements

The requirements for this procedure are a good random number source and working LF TRP link between BS and Key-Fob, as well as the ability of determining if a collision in the Key-Fob-BS communication happened.

A random number source is considered good or fitting, if it delivers a probability distribution close to unit distribution over an interval of eight. As this would deliver a minimum collision probability for choosing one out of the eight possible timeslots.

5.5 Pairing Procedure with Deterministic Collision Management

As this procedure is way more complex and implements a lot of logic and used data structures, each part will be explained separately in the following section and then a final overview will be given.

5.5.1 Generating the the group_ID

The group_ID is generated by the BS in order to request a special group of Key-Fobs; therefore its length is 32 bit and the group_ID does not need to be unique among the Key-Fobs. In figure 5.4 it can be seen, that depending on the iteration just a few of those 32 bits are actually used for the group_ID.

Imagine three Key-Fobs within range of the BS:

	ID
Key-Fob 1	011 010 011
Key-Fob 2	100 001 011
Key-Fob 3	101 001 011

Table 5.7: Key-Fob IDs.

Now the group_ID for each Key-Fob depending on the iteration is defined as follows:

Iteration	group_ID Key-Fob 0	group_ID Key-Fob 1	group_ID Key-Fob 2
0	000 000 000	000 000 000	000 000 000
1	000 000 011	000 000 011	000 000 011
2	000 010 011	000 001 011	000 001 011
3	011 010 011	100 001 011	101 001 011

Table 5.8: group_IDs corresponding to the iteration and the Key-Fob ID.

It can be seen that the group_ID basically is a masked version of the Key-Fob ID, with the mask depending on the iteration. Also in iteration 0, all Key-Fobs have the same group_ID, which is all zero.

5.5.2 Relation between Key-Fob ID and Reply-Timeslot

As long as the Key-Fob is not assigned to a fixed timeslot, the timeslot for replying to the BS is chosen dynamically. This timeslot is called reply-timeslot. The Key-Fob therefore uses 3 bits out of the unique 32 bit Key-Fob LFTRX SoC ID. Those three bits depend on the iteration, whereas always bit $[iteration \cdot 3]$ till bit $[iteration \cdot 3 + 2]$ are used. This is illustrated in figure 5.2 as well. In table 5.9 it is shown which reply-timeslot would be chosen by a Key-Fob with the ID 010 000 101 111 011, depending on the iteration:

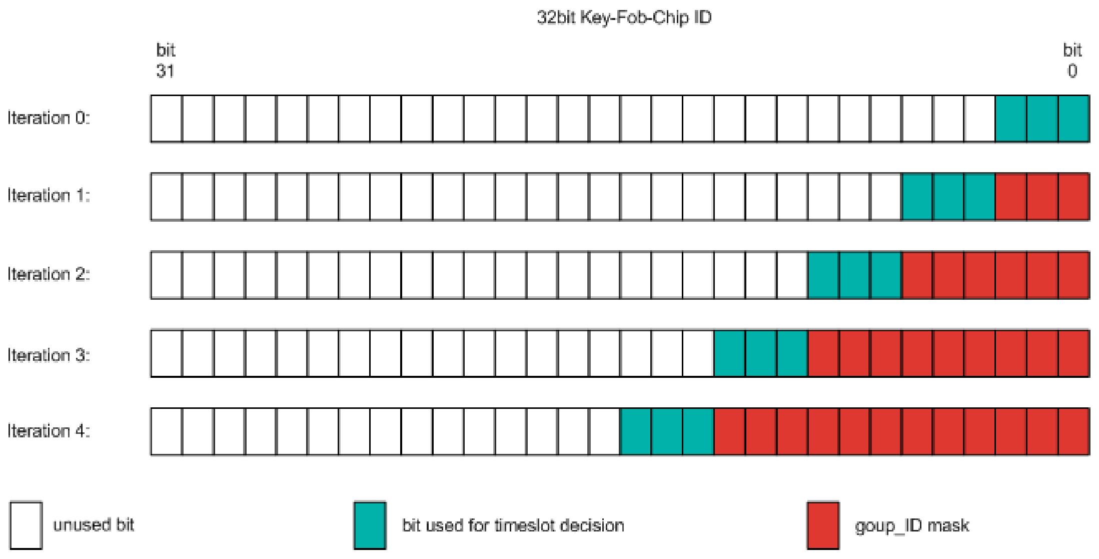

Figure 5.4: Usage of the ID bits depending on the iteration.

Iteration	Reply-Timeslot
0	$011 = 3$
1	$111 = 7$
2	$101 = 5$
3	$000 = 0$
4	$010 = 2$

Table 5.9: Chosen reply-timeslot depending on the iteration and the Key-Fob ID 010 000 101 111 011.

5.5.3 Relation between Collision-Timeslot and group_ID/Key-Fob ID

As seen in the sections above, there is a close relation between reply-timeslot and group_ID as well as the Key-Fob ID. In case of a collision, the reply-timeslots of multiple Key-Fobs are equal. This means that from the timeslot number and the iteration, the BS can calculate the group_ID, respective parts of the Key-Fob ID, of all Key-Fobs involved in the collision. Imagine following example; for the example the length of the group_ID and Key-Fob IDs is assumed to be 12 bit:

Iteration	timeslot valid replies	timeslot collision	group_ID	estimated Key-Fob IDs
				A: `XXXXXXXXX010`
0	–	2	000 000 000 010	B: `XXXXXXXXX010`
		4	000 000 000 100	C: `XXXXXXXXX100`
				D: `XXXXXXXXX100`
	1			A: `XXXXXX001010`
1	3			B: `XXXXXX011010`
		7	000 000 011 100	C: `XXXXXX111100`
				D: `XXXXXX111100`
				A: `XXXXXX001010`
2	–			B: `XXXXXX011010`
		0	000 000 011 100	C: `XXX000111100`
				D: `XXX000111100`
				A: `XXXXXX001010`
3		–	–	B: `XXXXXX011010`
	5			C: `101000111100`
	6			D: `110000111100`

Table 5.10: group_IDs and estimated Key-Fob IDs depending on replies and collisions in certain timeslots.

As the example in table 5.10 shows, the BS can learn the Key-Fob IDs as well as the group_IDs from the timeslot number in which collisions occur or replies are detected. After iteration 1, Key-Fob A and Key-Fob B are correctly detected. This means they are silenced (SEND_LOCK is set to 1) and will not reply to further commands. Therefore nothing can be learned about their ID, but this is not necessary anymore as the ID can simply be requested as LF packet from those Key-Fobs, using the first group_ID, recovered in iteration 0.

5.5.4 Formal Description of the Algorithm

The BS first sends a broadcast request (iteration = 0, group_ID = 0) to force all Key-Fobs to reply. Now it detects and saves all replies and saves all collisions, as the Key-Fobs choose their reply-timeslot as explained in section 5.5.1, with the timeslot number it occurred in, as well as all the correct received replies from Key-Fobs. All regularly detected Key-Fobs are assigned to a permanent timeslot and silenced, in order to keep them from interfering with the further detection process.

All collisions are then sequentially resolved using the iterations and group_ID to divide up the group of colliding Key-Fobs till only one Key-Fob per timeslot is left in the group and can therefore be detected correctly. New discovered collisions are also saved and later on resolved, until no collisions are left and all Key-Fobs in range of the BS got assigned to a permanent timeslot. All Key-Fobs are silenced now; therefore another broadcast request is send by the BS in order to check if there are any additional Key-Fobs left in range. If so, the same procedure as above is applied, if not the detection process and collision management is done.

As a last step of the pairing procedure the credential exchange can be started, to pair BS (respective the car) and Key-Fob. For that all Key-Fobs are un-silenced, so they reply to commands send by the BS.

5.5.5 Base Station Functionality

The BS controls the system according to the evaluation of the communication packets and the command replies.

Base Station Commands

Command	Parameter	Descriptions
REQUEST	iteration ID	Requests a certain Key-Fob or a group of Key-Fobs specified by the ID and the iteration-number.
SET_TIMING	ID timeslot number	Sets the timeslot number and SEND_LOCK for a certain Key-Fob specified by the ID.
START_CRED_EXHANGE	ID	Start credential exchange between BS and a certain Key-Fob or a group of Key-Fobs specified by the ID. Address 0 is a broadcast.
CLEAR	void	Pairing failed - reset all configuration registers to a reuse of the Key-Fobs.

Table 5.11: Description of the commands used by the BS.

Base Station Datastructures

Data Structure	Descriptions
Timing Table	Maps the IDs of the successfully identified Key-Fobs to the later on used timeslots.
Reply Table	In this table the replies of the Key-Fobs to a REQUEST command are saved. Therefore a mapping from reply timeslot to reply packet (ID and checksum) can be made.
Collision Stack	On this stack all collision are saved. One stack-entry consists of the tuple (group_ID, iteration). Whereas group_ID specifies which ID should be used collision and to request the Key-Fobs involved in this iteration the number of iterations already performed on this collision, incremented by one.

Table 5.12: Datastructures held by the BS.

Timing Table For further information see section 5.4.2.

Reply Table For further information see section 5.4.2.

Collision Stack The Collision Stack is used to track all the collisions that have occurred. One entry consists of iteration and group_ID, that needs to be used in the REQUEST command to make all Key-Fobs replies that caused this collision. An overflow of the Collision Stack is not possible with a proper HW setup, as the number of collisions that have to be saved at one time is limited. If the setup is wrong and an collision stack overflow occurs, the procedure will terminate and report an error.

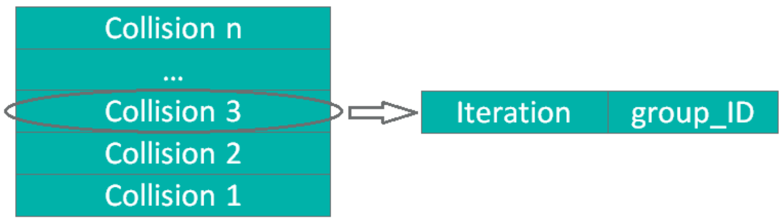

Figure 5.5: Structure of the collision stack and one collision stack entry.

Connection of the Datastructures

After a REQUEST command Key-Fobs will reply and the Reply Table will be filled with Key-Fob replies, according to the timeslot it was received in. Now the BSLFTRX iterates over all Reply Table entries and processes the entries. The following actions will be taken, depending on the Reply Table Entry:

Reply Table Entry Empty: No action taken

No collision detected in Reply Table Entry - Checksum OK Create an Entry in the Timing Table with the corresponding Key-Fob ID.

- Find free entry in Timing Table
- Set Timing Table entry “Key-Fob ID” to current Reply Table entry “reply Key-Fob ID”

Collision detected in Reply Table Entry - Checksum NOK: Push a new element on the Collision Stack. The “current iteration” and “current request ID” are held by the BSLFTRX.

- Collision Stack entry “Iteration” is set to “current iteration” incremented by one.
- Collision Stack entry “request ID” is set to
(“current request ID” | ((Reply Table “timeslot number”) << (3· “current iteration”))

Base Station State Diagram and Flowcharts

States The BS implements a finite state machine and operates in 4 different states (see figure 5.6) which will be explained in detail below.

INIT In the INIT state, the BS sends out a REQUEST command with iteration 0 and ID 0, which means that all Key-Fobs in range will send replies. Those are saved in the Reply Table and processed (see figure 5.7), so all successfully detected Key-Fobs are saved to the Timing Table and all collisions on the Collision Stack, according to the algorithm-specification. If those actions are completed without any error, the BS is set to state HANDLE_COLLISIONS, otherwise it is send to DONE.

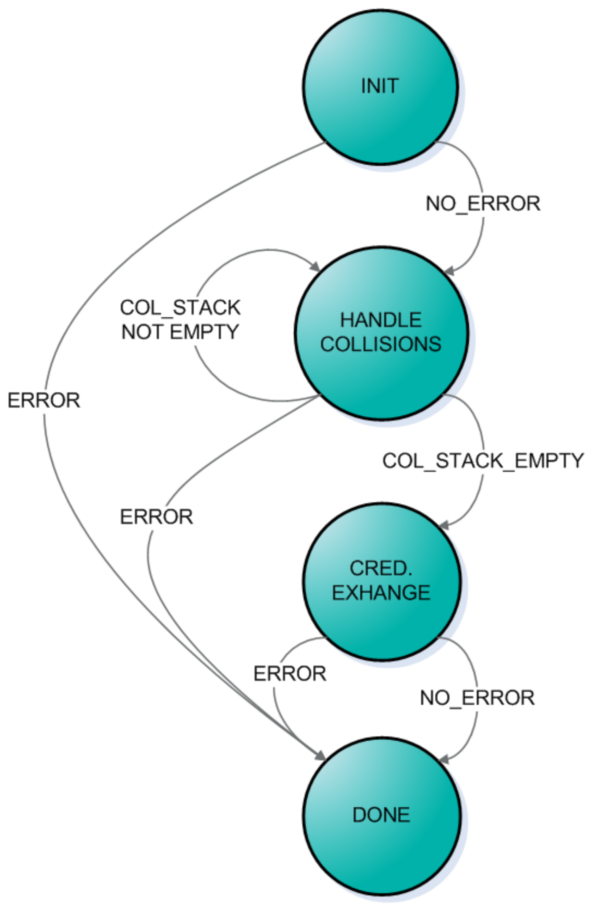

Figure 5.6: States of the BS in the collision management procedure.

HANDLE_COLLISIONS The BS stays in this state as long as there are elements on the Collision Stack. If all collisions are resolved, there is again a REQUEST broadcasted, to check if any Key-Fob has been missed, due to the dominant sender problem. The detailed program-flow in this state can be seen in figure 5.8.

CREDENTIAL_EXCHANGE In this state the BS starts exchanging the credentials with all registered Key-Fobs in the Timing Table, in order to complete the pairing.

DONE If any error occurred during the procedure, it is handled in this state. If the procedure finished without any error, the Timing Table is returned to the BSLFTRX main routine and the pairing procedure is terminated.

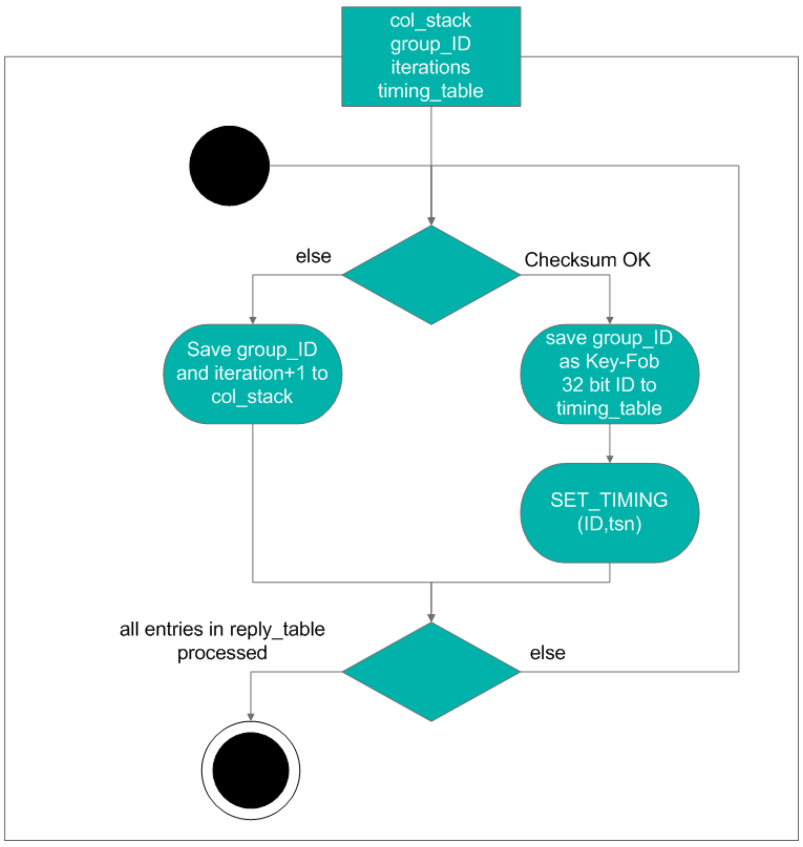
col_stack
group_ID
iterations
timing_table
else
Checksum OK
Save group_ID and iteration+1 to col_stack
save group_ID as Key-Fob 32 bit ID to timing_table
SET_TIMING (ID,tsn)
all entries in reply_table processed
else

Figure 5.7: Flow executed on the reply table (Process reply table).

5.5.6 Communication Sequence

Two Key-Fobs - No Collision

The sequence diagram (figure 5.9) illustrates the communication flow for a pairing process, considering two Key-Fobs in range of the BS. Those two Key-Fobs have 32 bit IDs which differ within the three LSB, which means that no collision happens after the initial REQUEST, send by the BS.

Three Key-Fobs - Collision

Figure 5.10 shows how the communication flow looks like for three Key-Fobs within range of the BS. The Key-Fobs 32 bit IDs do not differ within the three LSB but the ID of Key-Fob A differs in bits $3-5$. Also the IDs of B and C differ in the bit interval from bit 6 to 8. This causes a collision after the initial REQUEST by all Key-Fobs, and a collision between B and C after the second REQUEST.

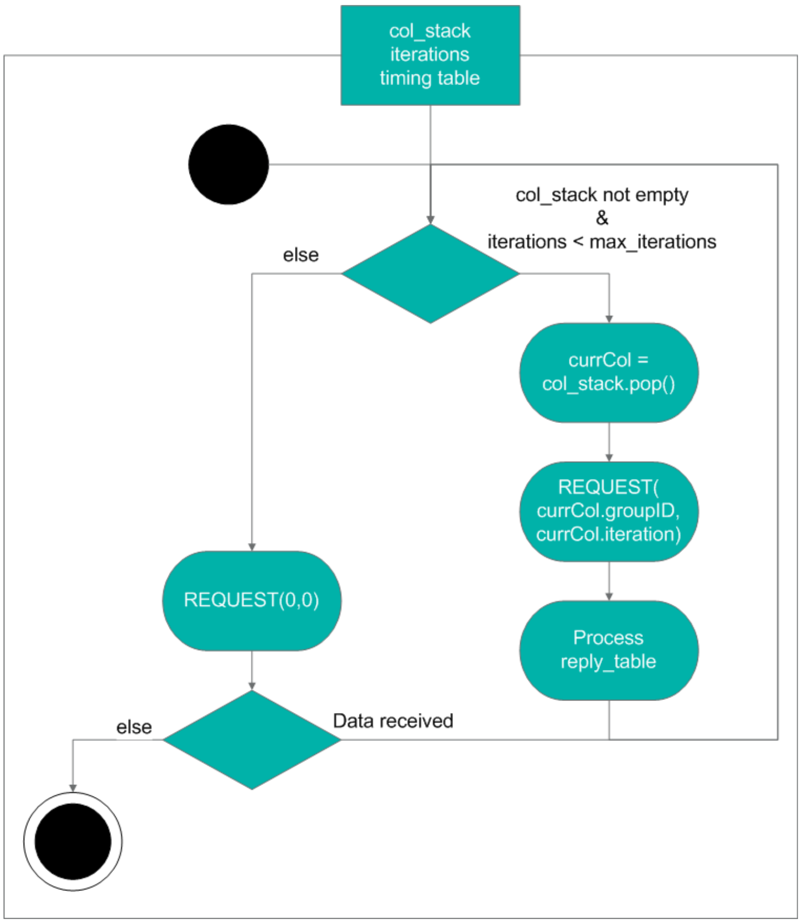

Figure 5.8: BS flow in the HANDLE_COLLISIONS state.

After the second REQUEST, Key-Fob A is successfully detected and the permanent timeslot as well as the SEND_LOCK is set for this Key-Fob. This means that Key-Fob A is not responding to any further REQUEST commands anymore. The collision between B and C is resolved, as described above, and then the permanent timeslots for those Key-Fobs can be set as well and the credential exchange can be started.

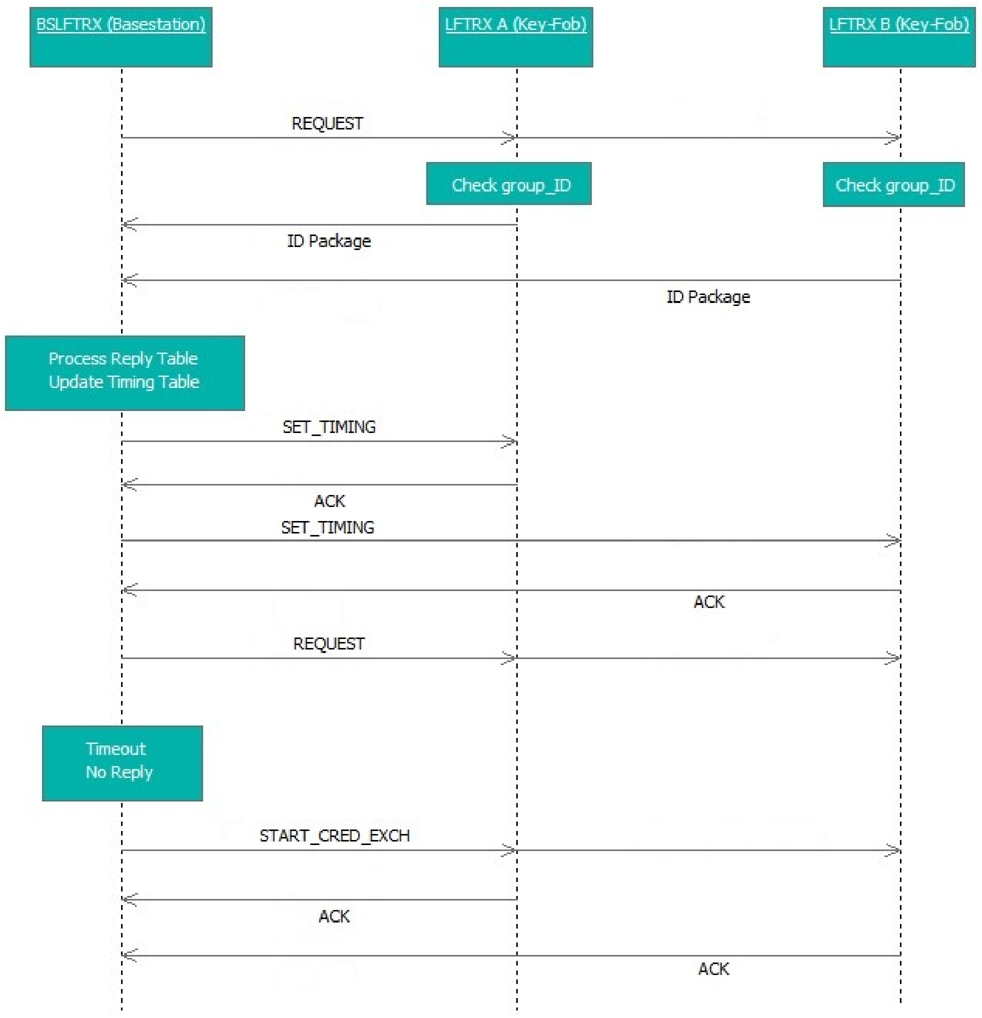

Figure 5.9: Sequence diagram for a complete communication flow between BS and two Key-Fobs; no collisions.

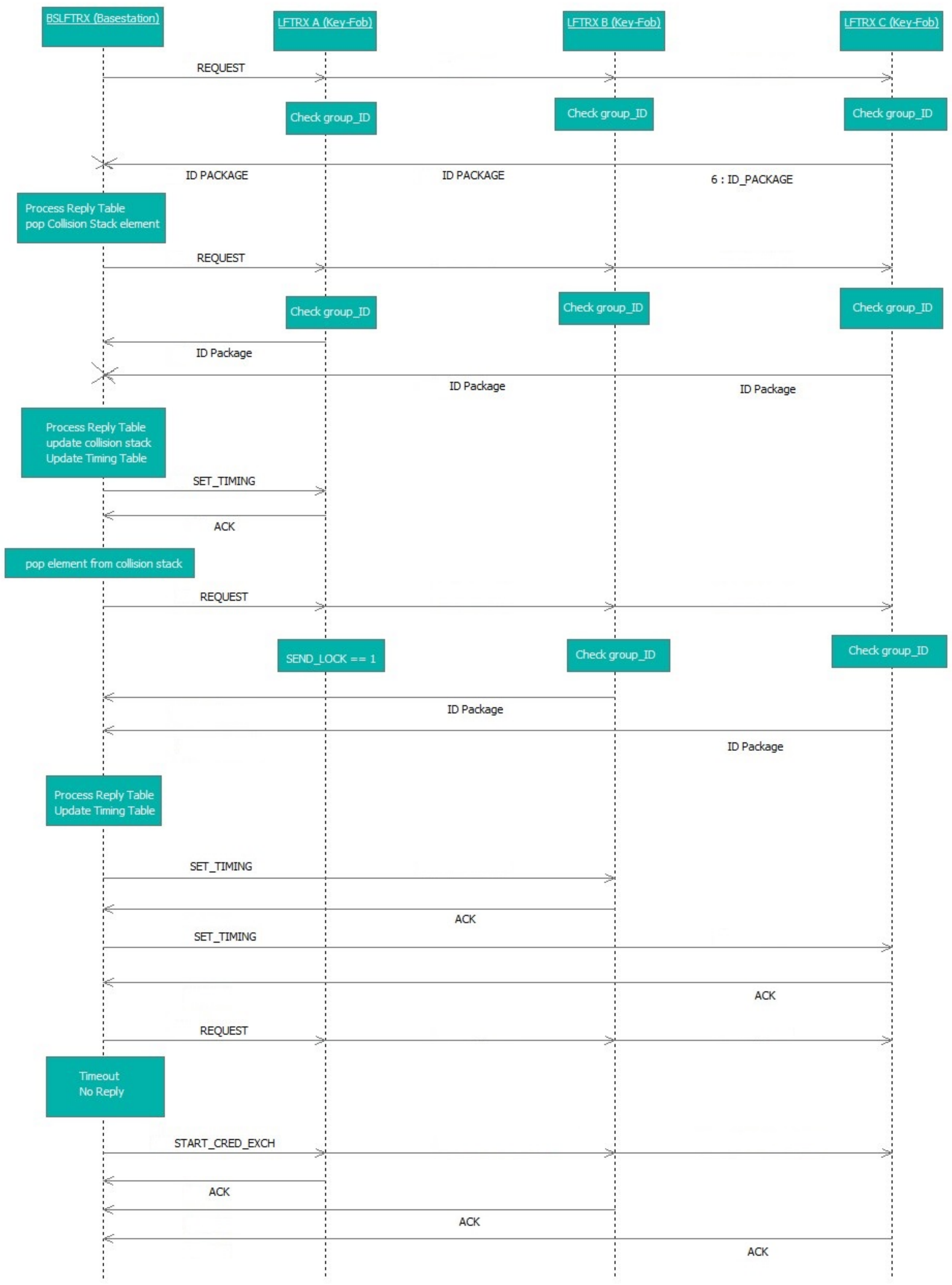

Figure 5.10: Sequence diagram for a complete communication flow between a BS and three Key-Fobs; collisions appear.

5.5.7 Example

Key-Fob Name	Key-Fob ID
A	111 110 011 010
B	101 000 010 010
C	100 010 011 101
D	101 110 011 010
E	100 010 110 101

Table 5.13: IDs of the Key-Fobs used in this example.

Initially the BS requests all keys by sending the REQUEST command with ID 0 and iteration 0 - which equals a broadcast request. Then the reply-table is filled with information according to the replies.

Timeslot	Reply Key-Fob ID	Checksum
0	–	–
1	–	–
2	XXXXXXXXXXXX	XXX
3	–	–
4	–	–
5	XXXXXXXXXXXX	XXX
6	–	–
7	–	–

Table 5.14: Reply Table after iteration 0.

Now all entries in the reply-table are processed and two entries are added to the collision stack for timeslot 2 and 5.

1	000 000 000 101
1	000 000 000 010
Iteration	group_ID

Table 5.15: Collision Stack after iteration 0.

After the initial round, the BS starts to resolve all collisions saved on the collision stack. Therefore the latest entry from the collision stack is popped and the BS sends a REQUEST with ID = 000 000 000 101 and iteration 1. After this the reply table looks like follows:

Timeslot	Reply Key-Fob ID	Checksum
0	–	–
1	–	–
2	–	–
3	100 010 011 101	XXX
4	–	–
5	–	–
6	100 010 110 101	XXX
7	–	–

Table 5.16: Reply Table after iteration 1 for the first collision entry.

As can be seen in the table 5.16, now two Key-Fobs have been detected in timeslot 3 (Key-Fob C) and in timeslot 6 (Key-Fob E). Those two Key-Fobs are silenced and get assigned to permanent timeslots 1 and 2. This means the Timing Table now looks as follows:

Timeslot	Key-Fob ID	Valid
0	100 010 011 101	1
1	100 010 110 101	1
2	–	0
3	–	0
4	–	0
5	–	0
6	–	0
7	–	0

Table 5.17: Timing Table after detecting Key-Fob C and E.

Now the remaining entry in the collision stack is popped and a REQUEST with iteration 1 and group_ID = 000 000 000 010 is send. The reply table then has following entries:

Timeslot	Reply Key-Fob ID	Checksum
0	–	–
1	–	–
2	101000010010	XXX
3	XXXXXXXXXXXX	XXX
4	–	–
5	–	–
6	–	–
7	–	–

Table 5.18: Reply Table after iteration 1 for the second collision stack entry.

Key-Fob B is now detected in timeslot 2 and will be assigned the permanent timeslot 3 and added to the Timing Table. For the collision detected in timeslot 3 a new entry in the Collision Stack is made containing (iteration, group_ID) = (2, 000 000 101 010). The BS will then pop the Collision Stack entry send a REQUEST with iteration = 2 and ID = 000 000 101 010 as specified in this entry, this will again result in a collision in timeslot 6. Therefore the entry inserted in the Collision Stack is now (iteration, group_ID) = (3, 000 110 101 010). Popping this entry and sending a REQUEST with iteration 3 and ID = 000 110 101 010 will then result in Key-Fob A replying in timeslot 7 and Key-Fob D in timeslot 5. Now those two Key-Fobs are assigned the permanent timeslots 4 and 5.

Timeslot	Key-Fob ID	Valid
0	100 010 011 101	1
1	100 010 110 101	1
2	101 000 010 010	1
3	101 110 101 010	1
4	111 110 101 010	1
5	–	0
6	–	0
7	–	0

Table 5.19: Final Timing Table.

As the Collision Stack is empty now, the BS sends again a REQUEST with iteration 0 and group_ID 0 to check if there are any Key-Fobs left in range. As this not the case and no reply is received, the BS broadcasts the START_CRED_EXCH which initializes the credential data-transfer. After successfully finishing the credential exchange, the procedure is finished and the routine terminates.

5.5.8 Proof of Correctness

The 32 bit LFTRX SoC ID is imagined like a tree, similar to figure 5.11. As the binary search algorithm is proven correct to be able to find all items in a tree, so is the Key-Fob selection used in this procedure.

The innovative part of this procedure is the selection of the timeslots, using the unique 32 bit ID of the LFTRX SoCs on the Key-Fobs. As the ID is considered unique, the following statements can be made:

- Considering two Key-Fobs with IDs that just differ in one bit, after a certain amount of iterations the difference either lies in the part of the ID that is used for the group_ID or the bits that are used to select the reply-timeslot.
- If the difference lies in the part used for the group_ID, only one of those Key-Fobs will reply to a REQUEST as the group_ID just fits for one of the two Key-Fobs. No collision possible.
- If the difference lies in the bits used for choosing the reply-timeslot, the Key-Fobs will reply in different timeslots and therefore no collision will occur.

Therefore all parts of the algorithm are proven correct which concludes the correctness of the algorithm.

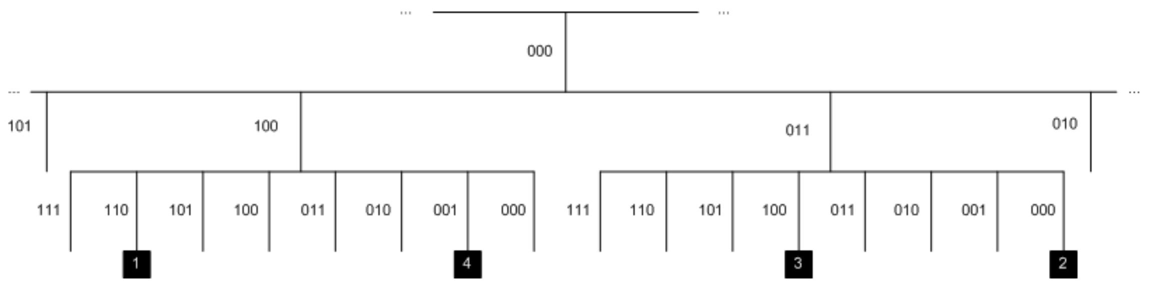

Figure 5.11: Part of a binary tree for the ID of the LFTRX SoC. ID1 = 000 100 110; ID2 = 000 011 000; ID3 = 000 011 100; ID4 = 000 100 001

5.5.9 Proof of Termination

Every LFTRX SoC ID is unique, therefore every Key-Fob will be found due to the correctness of the binary search algorithm and the theorem that Key-Fobs either differ in group_ID or timeslot selection bits. If all Key-Fobs are found the algorithm terminates.

5.5.10 Run Time Analysis

In order to calculate the probable runtime of the procedure a MATLAB script has been developed.

Calculate the initial probabilities for a collision

In the first iteration the first three bits (bit 0 - bit 2) are used by all Key-Fobs to determine the reply timeslot. Therefore we divide the total number of Key-Fobs N into subgroups of the size $\frac{N}{G}$, all Key-Fobs with equal bits 0 - 2 are gathered in one subgroup, and therefore will result in choosing the same reply timeslot which causes a collision. G here represents the number of timeslots (8). So, if more than one Key-Fob from a subgroup is chosen in our set of 8 Key-Fobs, there will be a collision in the initial round.

This can be described using a hypergeometric distribution. A collision is considered a success, therefore in a total population of N Key-Fobs there are $\frac{N}{G} = K$ Key-Fobs that produce a collision and therefore a success if they are drawn. Key-Fobs taken from the total population are not replaced.

Calculation of the first layer of the iteration tree

In one step a collision can be resolved, partly resolved or not resolved. Therefore following probability trees of depth 1 are now calculated to build the base for the final probability tree (see figure 5.12).

Now the total population of Key-Fobs is defined by the size of the collision (init_col_sz = [2 : 8]). There are $G^{\text{init\_col\_sz}}$ possibilities to distribute the Key-Fobs, that produced a collision in the last iteration, over the G (= 8) timeslots in this iteration.

The number of good distributions is that one timeslot is chosen by res_col_sz keys (number of keys that will generate a collision in this iteration) and the other (init_col_sz - res_col_sz) Key-Fobs are distributed over the other $G - 1$ groups, not caring about their specific distribution.

Therefore the probability to get from a collision of size init_col_sz to res_col_sz is

$$\#[\text{Possible timeslots for res\_col\_sz Key-Fobs to send back}] = \binom{G}{1}$$

$$\#[\text{Possibilities distribute remaining Key-Fobs over } G-1 \text{ timeslots}] =$$
$$= (G-1)^{\text{init\_col\_sz - res\_col\_sz}}$$

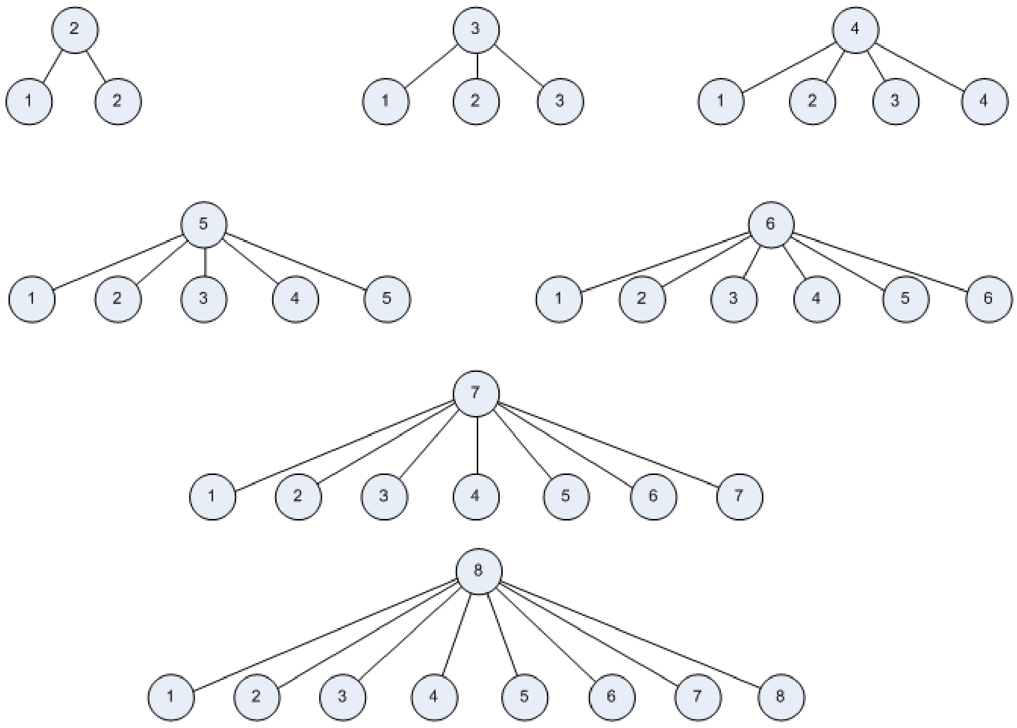

Figure 5.12: Collision Trees for all possible collision sizes till a depth of 1 (1 iteration). The numbers indicate the number of Key-Fobs replying in the same timeslot (res_col_sz).

$$\begin{aligned}\text{P\_tree(init\_col\_sz, res\_col\_sz)} &= \frac{\#\text{[good cases]}}{\#\text{[possible cases]}} \\ &= \frac{\binom{G}{1} \cdot (G-1)^{\text{init\_col\_sz - res\_col\_sz}}}{G^{\text{init\_col\_sz}}} \\ &= \frac{G \cdot (G-1)^{\text{init\_col\_sz - res\_col\_sz}}}{G^{\text{init\_col\_sz}}}\end{aligned}$$

Probabilities to resolve the Collision after a certain Number of Iterations

To calculate the probability of resolving a collision (of certain size) in a defined number of iterations, all the possible paths from the root of a collision tree to the leaves in depth iterations, are calculated and added up (example in figure 5.13).

Example: Initial collision size of 4; Resolving after 3 iterations

$$\begin{aligned}\text{P\_term}(3,4) = &\text{P\_tree}(2,4) \cdot \text{P\_term}(2,2) + \\ &\text{P\_tree}(3,4) \cdot \text{P\_term}(2,3) + \\ &\text{P\_tree}(4,4) \cdot \text{P\_term}(2,4)\end{aligned}$$

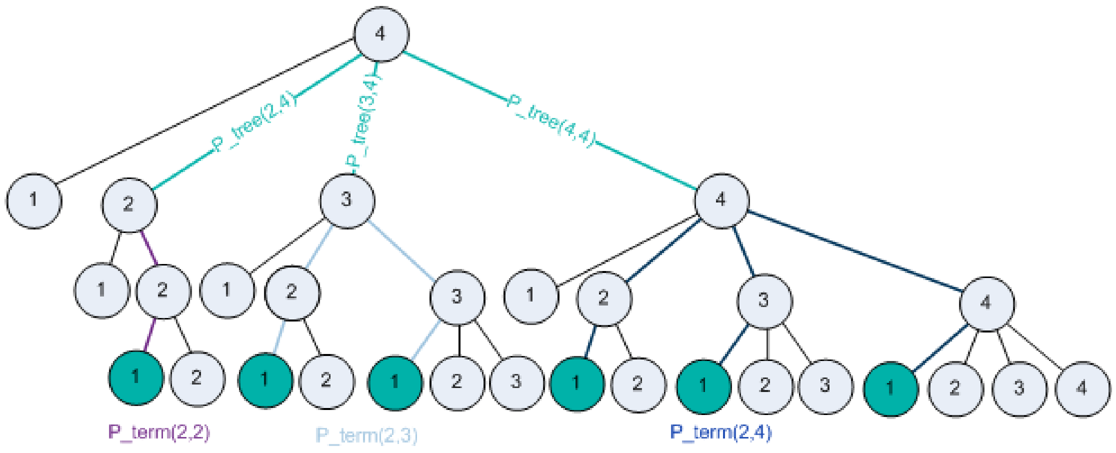

Figure 5.13: Collision Tree for a initial collision of 4 and a termination at iteration 3.

Final Calculation

Now all the termination probabilities are scaled with the probability to have a collision of certain size in the initial step. Finally all the probabilities, to detect a Key-Fob after a certain number of iterations for all possible collision patterns in the first iteration, are added up to get the total probability for each number of iterations.

Results

If one Key-Fob is observed within the system consisting of eight Key-Fobs, figure 5.14 shows the probability to detect this Key-Fob within a certain number of iterations. Iteration 0 represents the initial step, where the first REQUEST broadcast is send to all Key-Fobs. The iterations 1 to n represent the iteration steps, needed to resolve a collision found in the initial step (iteration 0).

As can be seen in the graph, the observed Key-Fob is detected in the initial step with a probability of 34.36%, which means this Key-Fob is not involved in a collision. This also means, that a observed Key-Fob is involved in a collision, in the initial step of the procedure, with a probability of 65.64%. If the Key-Fob is involved in a collision, within iteration 0, it will be detected after iteration 1 with a probability of 61.49% and after iteration 2 with 3.497%. This means, a observed Key-Fob in our system, is detected within 3 steps (iterations 0 to 2), with a probability of 99.34%.

Applying this data, to the whole system of eight Key-Fobs, this means that it is very likely that every Key-Fob is found within 3 steps. Considering the maximum number of 4 collisions of size 2 in iteration 0, this would lead to a duration of 9 steps (4x 2 iterations

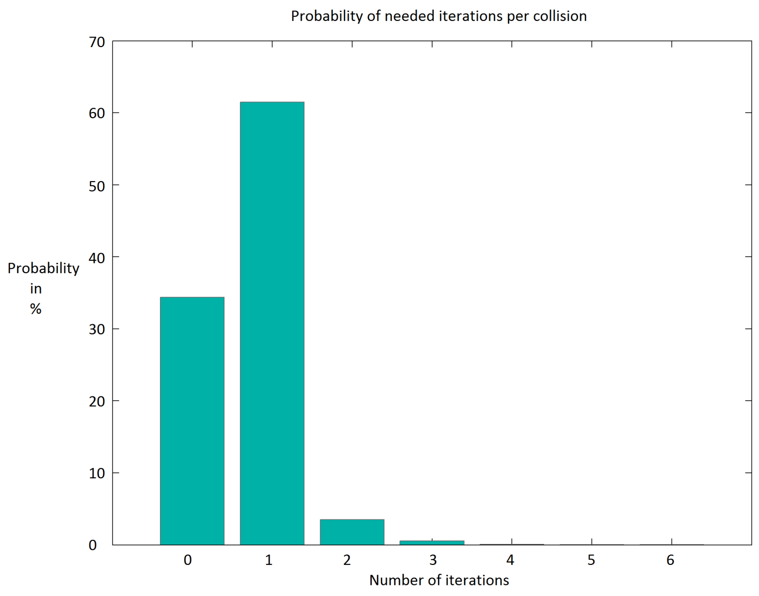

Figure 5.14: Probability of the collision management procedure terminating after a certain number of iterations for a single Key-Fob.

for collision resolving plus iteration 0), as for every resolved collision, 2 Key-Fobs are found. Therefore it can be assumed, that for most of the possible setups, the procedure will terminate after 1 to 9 steps.

5.5.11 Requirements

The prerequisites for this procedure are that all the Key-Fob LFTRX SoCs have a unique 32 bit ID and a working LF TRP link between Key-Fob and BS, as well as the ability of determining if a collision in the Key-Fob-BS communication happened.

Chapter 6

System Analysis and Constraints

6.1 Requirements and Constraints Analysis for the Pairing Process

6.1.1 Pairing Trigger

Defining the trigger for the pairing procedure is up to the OEM, therefore the procedure is kept as flexible as possible. In this initial version, the procedure is designed to be able to be triggered multiple times. Adding or deleting of single Key-Fobs is not possible for this version, therefore every Key-Fob has to be present in every pairing run.

Furthermore it is assumed that the procedure can just be triggered within a secure environment (see section 6.1.2) which directly influences the security features of the system.

6.1.2 Security

As this system is dealing with security critical components, one of the focus points is security. In the first implementation of the pairing procedure with collision management, no security measures will be added to the system due to following considerations:

- The range of the LF TRP communication is approximately 10 cm, therefore sniffing would just be possible within the car.
- As stated in section 6.1.1, the pairing procedure can only be triggered in a secure environment. Therefore an attacker is considered to not be able to get into the car.

For future developments of the system, advanced security measures should be implemented in order to make the system harder to attack and the procedure also usable in a non-secure environment.

Secure Environment

An environment is considered to be secure, if it is highly unlikely that an attacker is able to get close enough to the system to be able to eavesdrop on the communication between the BS and the Key-Fobs. Also all environmental factors are within the operation ranges of the system (e.g. supply voltage is sufficient). Such an environment is for example the factory of the car-manufacturer or a licensed car-shop.

6.1.3 System Setup

Unexpected Number of Key-Fobs

The system maximally supports eight Key-Fobs with one BS. Therefore it is up to an operator to place a maximum of eight Key-Fobs in range of the BS before the pairing procedure is triggered. If the operator fails to do so, following cases can occur

i) No Key-Fobs in range

ii) More than eight Key-Fobs in range

If no Key-Fobs are in range the procedure will terminate without having paired with any Key-Fob. If there are too many Key-Fobs in range, there will be Key-Fobs recognized in range although the timing table is already full. In this case the BS will abort the pairing procedure, reset all Key-Fobs that were already registered and report an error to the car-electronics. This way the operator can check the number of Key-Fobs in range and trigger the pairing procedure again.

Change of System Setup during the pairing process

Considering the pairing trigger constraints, it is it is assumed that the setup is not changing during the pairing procedure (removing/adding of a Key-Fob).

Multiple Basestations

As the range of the LF TRP communication is maximum 10 cm and the triggering of the procedure is assumed to be happening in a secure environment, it is highly unlikely that a second BS is introduced to the system. Therefore this case is not considered in the implementation.

Variations in Supply Voltage

Variations of the supply voltage of the BS highly influences its functionality, in terms of communication range. Also a total fail of the supply voltage could occur. Those scenarios are considered to be highly unlikely considering the triggering of the procedure defined in section 6.1.1.

Moreover, the supply of the Key-Fob will not run out as long as the supply of the BS is given, as the Key-Fob harvests the operating energy out of the LF field in TRP mode and is therefore not dependent on any other power supply (e.g. battery).

Checksum Collision

A collision that is not recognized due to a faulty checksum is highly unlikely to occur. If so, it will lead to the termination of the procedure with an error code.

6.1.4 Verification of the Procedure

The first implementation of the pairing procedure does not need a detailed verification (e.g. challenge-response for every Key-Fob) with an error handling. A simple Acknowledge (ACK) is sufficient due to the assumptions taken in the sections above.

6.1.5 Requirements for the Pairing Procedure

Collisions detection

In order to implement a collision management algorithm, it is necessary to be able to detect collisions. As defined in section 5.2, a Fletcher-16 checksum is included in the ID-Packets sent from the Key-Fob to the BS after a REQUEST. Therefore a data integrity check can be performed, which shows collisions if the data does not fit the checksum.

Furthermore, as mentioned in literature, also the dominant sender problem can occur. This means that the signal from a client is that much stronger than the signal of all other clients, so the host cannot detect a collision and just sees one client. Tests, described in appendix A, showed that this problem can also occur in the KGO System. Therefore both theoretical algorithms described in chapter 5 implement a final check, to avoid missing any Key-Fobs in range due to the dominant sender problem.

Random Number Sources

In order to implement the pairing procedure with stochastic collision management, a good random number source is needed. In the appendix A all possible random sources on the LFTRX SoC, which are available in TRP mode, were analysed. The results showed, that those sources are not sufficient and do not deliver any randomness.

Also it has been considered to use pseudo-random numbers, by XORing and other mathematical calculations. This has been discarded due to the low computational power of the LFTRX SoC, as a simple procedure would most likely result in many collisions, and a complex procedure is not feasible.

32 bit Key-Fob LFTRX ID

The pairing procedure with deterministic collision management requires an unique Key-Fob LFTRX ID. It is known that the LFTRX SoC has a unique 32 bit ID. Also considering the number of cars produced per year, and the ascending numbering of the Key-Fob IDs, it is highly likely that the IDs will differ within a shorter range of bits than 32. Also, in the first implementation of the procedure, the case of multiple Key-Fobs with identical IDs is not considered. Furthermore, it is not possible that a LFTRX SoC in the Key-Fob changes its ID, as this ID is permanently programmed in the fab.

6.2 Conclusion of the Algorithm Analysis

The analysis shows that the stochastic approach is less complex than the deterministic approach and guarantees termination and proper functionality, if there is a proper random number source available. As section A in the appendix shows, apparently there is no proper random number source available at the LFTRX SoC, and therefore this procedure cannot be used in the KGO System.

Therefore the deterministic approach is implemented, due to its guaranteed good performance (see section 5.5.10) and the guaranteed termination. It has to be taken into account that this solution is less flexible as it is only tailored for the KGO System and the SoCs used in it.

Chapter 7

Implementation, Test and Verification of the Procedure

In chapter 5 the pairing procedure with collision management was theoretically designed and analysed. This chapter focuses on the implementation, test and verification of this procedure within the context of the KGO System. Again it should be mentioned, that the procedure defined in section 5.5 was chosen, due to the reasons stated in chapter 6.

7.1 Firmware Development within the Context of the Keyless Go System

Besides the HW for the KGO System, Maxim Integrated also offers parts of the FW needed to operate the SoCs, used in the system. As figure 7.1 illustrate, the FW is build in a layer structure. The FW developed in the context of this master thesis will also take advantage of the existing layers, and can be considered part of the application layer.

As the figure 7.1 shows, the FW design consist of 4 layers, which are defined as follows from bottom up:

- Layer 0:
  The Low Level Drivers form the HW Abstraction Layer. It mainly consists of register access macros and bit masks for special SoC functionality and configurations.
- Layer 1:
  The basic setup and control functions are gathered in the Physical Layer. Those functions just implement one simple setup for a functional block.

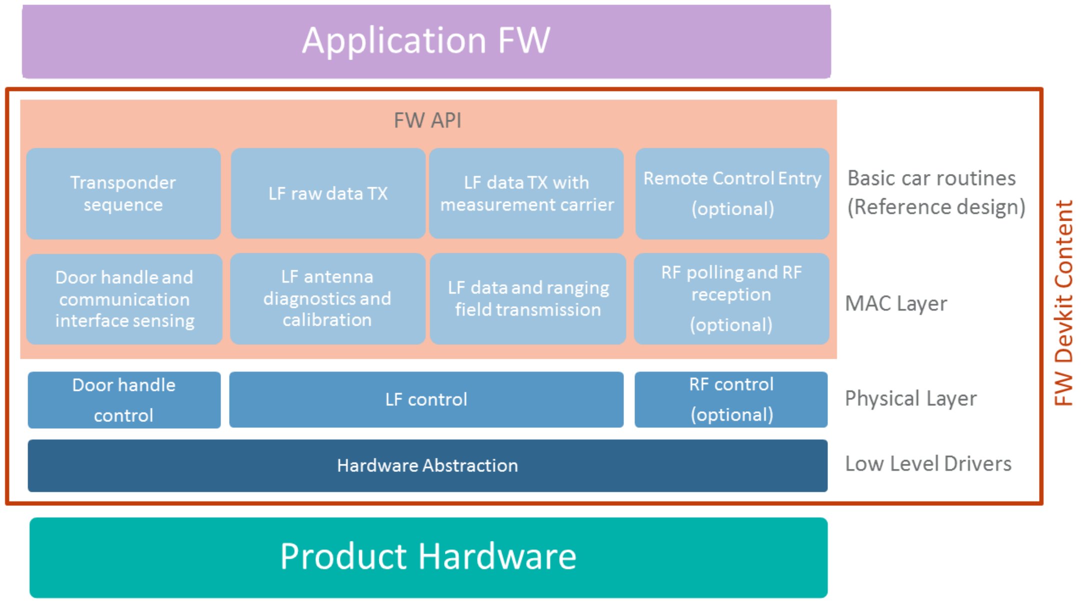
Application FW
FW API
Transponder sequence
LF raw data TX
LF data TX with measurement carrier
Remote Control Entry (optional)
Basic car routines (Reference design)
Door handle and communication interface sensing
LF antenna diagnostics and calibration
LF data and ranging field transmission
RF polling and RF reception (optional)
MAC Layer
Door handle control
LF control
RF control (optional)
Physical Layer
Hardware Abstraction
Low Level Drivers
FW Devkit Content
Product Hardware

Figure 7.1: FW structure for the KGO System. Left the structure for the Key-Fob unit, right for the BS.

- Layer 2:
  This layer represents the FW Application Programming Interface (API) and consists of two sub-layers:
  - The functions in the MAC Sub-Layer combine multiple functionality from single functional blocks.
  - The Basic-Routines sub-layer contains already usable flows and procedures. Although this layer should not implement any protocols specific flows which can be redefined by a customer.
- Layer 3:
  The highest level is the Application FW layer. Here customer specific flows and protocols are implemented as well as customer application FW.

Maxim Integrated provides the Layers 0 to 2 already fully tested and verified. The implementation of the pairing and collision management procedure is contained in Layer 3.

7.2 Code Design

7.2.1 Base Station LF Transceiver SoC

In order to ensure high re-usability and easy maintenance the FW is designed modular with as few interfaces to the SoC specific functionality as possible. The structure can be seen in figure 7.2.

The modules are described in detail in the paragraphs below.

SoC main

This is the main routine of the FW, it handles commands received via the wired interface and also implements some debug and configuration commands in order to set up the Key-Fobs for specific test-cases. Here also the pairing and collision management procedure is triggered by calling PAIR_main, if a specific command is received.

PAIR_main

PAIR_main is the top function and implements the state-machine as illustrated in figure 5.6.

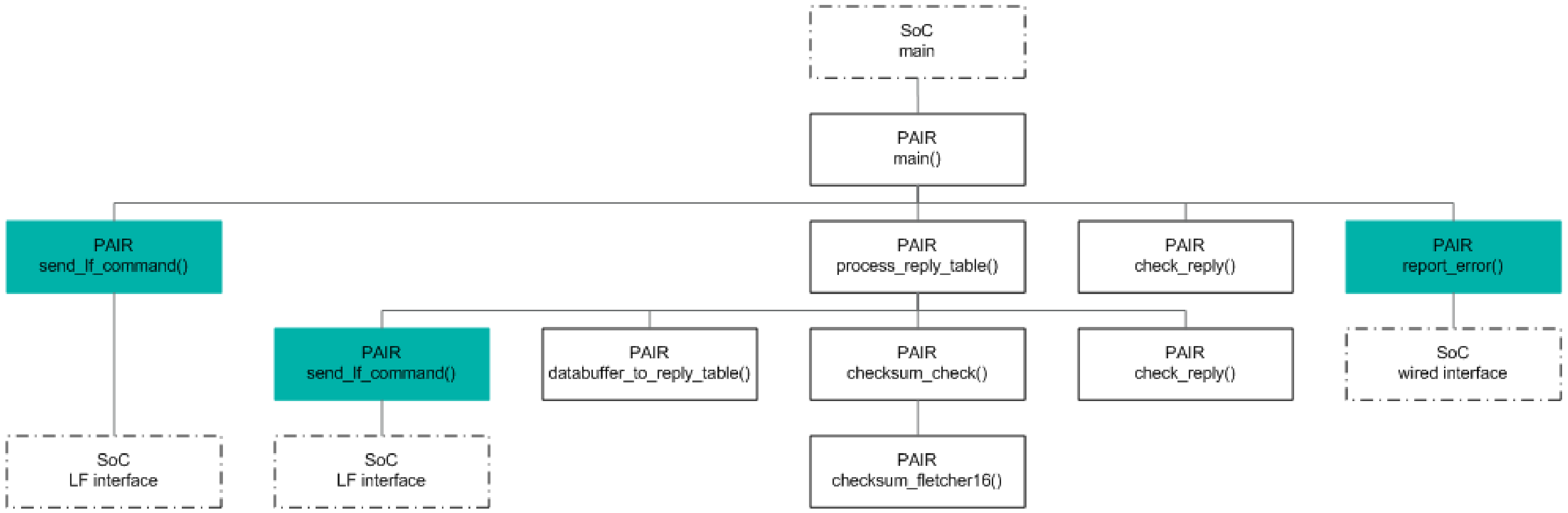

Figure 7.2: FW structure of the algorithm implementation on the BS, the colored blocks use BSLFTRX SoC specific functions and the blocks with hatched borders represent the SoC functionality.

PAIR_send_lf_command

In order to separate the call of all LF interface SoC functions, they are gathered in this function. The function takes care of the LF setup, assemble of the data packet as well as sending and receiving of the data by calling the relating SoC specific functions.

PAIR_process_reply_table

This function processes the data in the reply table as described in section 5.5.5 in paragraph *Connection of the Datastructures*.

PAIR_check_reply

This function simply checks if the received function code and data match the expected values.

PAIR_report_error

When the procedure terminates, this function assembles a report packet according to the termination error_code, which is send to the host via the wired interface.

PAIR_databuffer_to_reply_table

As the data received via the LF link is just stored byte-wise in a buffer, it must be pre-processed and saved in the right format in the Reply Table. Therefore this function separates the received data for every timeslot into function code, Key-Fob reply ID and Checksum and saves it to the Reply Table into the according slot.

PAIR_checksum_check

In order to check if a packet is valid, this function uses the PAIR_checksum_fletcher16 function to calculate the 16 bit Fletcher Checksum over specific data, and compares it to the received checksum.

PAIR_checksum_fletcher16

This function is used to compute the 16 bit Fletcher Checksum.

SoC wired interface and SoC LF interface

Those blocks represent the BSLFTRX SoC specific function to setup and use the wired (either 3WI or UART) and the LF interface4.

7.2.2 Key-Fob LF Transceiver SoC

As the LFTRX in the Key-Fob functionality consists just of event driven operations and due to the low computational power of the LFTRX SoC the FW is held as simple as possible. Therefore it consists just of two modules:

- Main Routine and Event handling
- Checksum Calculation (Fletcher)

Firmware limitations due to memory restrictions

As all of the memory except the Flash is volatile, permanent information like the timeslot number have to be stored there. When using the Flash in the LFTRX following things need to be considered:

- The Flash can just be written in words, whereas a word-size is 16 bits.
- Reading from the Flash can be done byte-wise.
- The Flash is initialized with all ones.
- The Flash just supports destructive writing[1].
- The Flash memory can just be reset through either a mass erase or a page erase. As also the initial configuration is stored in the Flash, a mass erase must not occur.

These facts also cause some modifications in the algorithm design. As there is no memory location to store the SEND_LOCK permanently which can be written multiple times, it will be derived from the permanent timeslot number, which is also stored in the Flash. If there is a valid timeslot number stored, the SEND_LOCK is set. Furthermore, if a CLEAR command is received by a Key-Fob, it does a page erase on the page which holds the timeslot number.

[1]Ones can just be set to zero, no writing of ones over zeros possible.

7.3 Conformance and Test-Case-Success

The test-cases check the conformance of the input and output. The input to the algorithm are the IDs, initial SEND_LOCKs and initial timeslot number of the eight Key-Fobs. This input is used to configure the whole environmental setup. The outputs are the error_code and timing table. Each test-case defines a input set of size 8x3, consists of 8 triples (ID, SEND_LOCK, timeslot number), and produces a output set of (error_code, timing table).

The conformance is then defined as follows:

$$\begin{aligned} i \ \ldots \ \text{Implementation} \qquad & x \ldots \ \text{Input Set} \\ m \ \ldots \ \text{Model} \qquad & X \ldots \ \text{Possible Input Sets} \end{aligned}$$

whereas

$$i \leq_{conf} m =_{def} \forall x \in X : \mathbf{out}\,(i \ \mathbf{for} \ x) ==_{out} \mathbf{out}\,(m \ \mathbf{for} \ x) \tag{7.1}$$

has to hold. For more detailed definitions and derivations see appendix B.

Therefore a test-case t with the input set x_t and is considered successful for a implementation i and the model m if

$$t = \text{successful } \mathbf{if} \ \mathbf{out}\,(i \ \mathbf{for} \ x) ==_{out} \mathbf{out}\,(m \ \mathbf{for} \ x) \tag{7.2}$$

7.4 Test-Plan

The testflow is divided in three main parts, Software (SW)-in-the-Loop tests, HW-in-the-Loop tests, and manual review. Those are defined as follows:

i) SW-in-the-Loop Test
   a) Static Code Analysis with *LINT*
   b) Manual Analysis
   c) Test-Case-Generation
      - Test-Case-Generation with CREST [BS08]
      - Manual test-case definition
   d) Model Test
   e) Generate Extensible Markup Language (XML)-test-case specifications

ii) HW-in-the-Loop Test
   a) Key-Fob Tests

b) System Tests

iii) Manual Review

Figure 7.3 shows the structure of the whole Test Framework, and each component is described in detail in the following sections. A detailed manual on how to setup the setup can be found in appendix C, the following sections will just give an overview and evaluate the testing.

7.4.1 Software-in-the-Loop

Manual Code Analysis

The code of the LFTRX and BSLFTRX SoCs needs to be manually examined beforehand. This has to be done to evaluate if its feasible to model and pre-process it, in order to use the automatic test-case-generator and the MATLAB test framework on this code. This decision is based on the amount of statements in the code, that requires SoC specific functionality, and are therefore hard to model, so they can be used in a SW loop.

Test-Case-Generator

For the automatic test-case-generation and code analysis, CREST (a concolic test generation tool for C [BS08]) was used. This is a simple command line tool for *UNIX* systems. The tool builds on the yices [DdM06] packet, a Satisfiability Modulo Theories (SMT) solver for constrain solving, and CIL [NMRW02], a tool to extract the control-flow graphs from C code. Concolic testing is the technique to execute a program under test during the test symbolic, as well as concrete. For this concolic testing, CREST supports several searching methods to explore the maximum number of branches in the program:

- Bounded Depth-First Search
- Control Flow Directed Search
- Uniform Random Search
- Random Branch Searching

Also random input generation can be done using CREST. Branch coverage is used as quality measurement for the tests generated with CREST. Using a better measurement, like path-coverage, is not possible. The advantages of CREST are the easy setup and fast compile for almost all *UNIX* distribution, but it has the big disadvantage that it can just be instrumented for single source file programs. Therefore, a script has to be

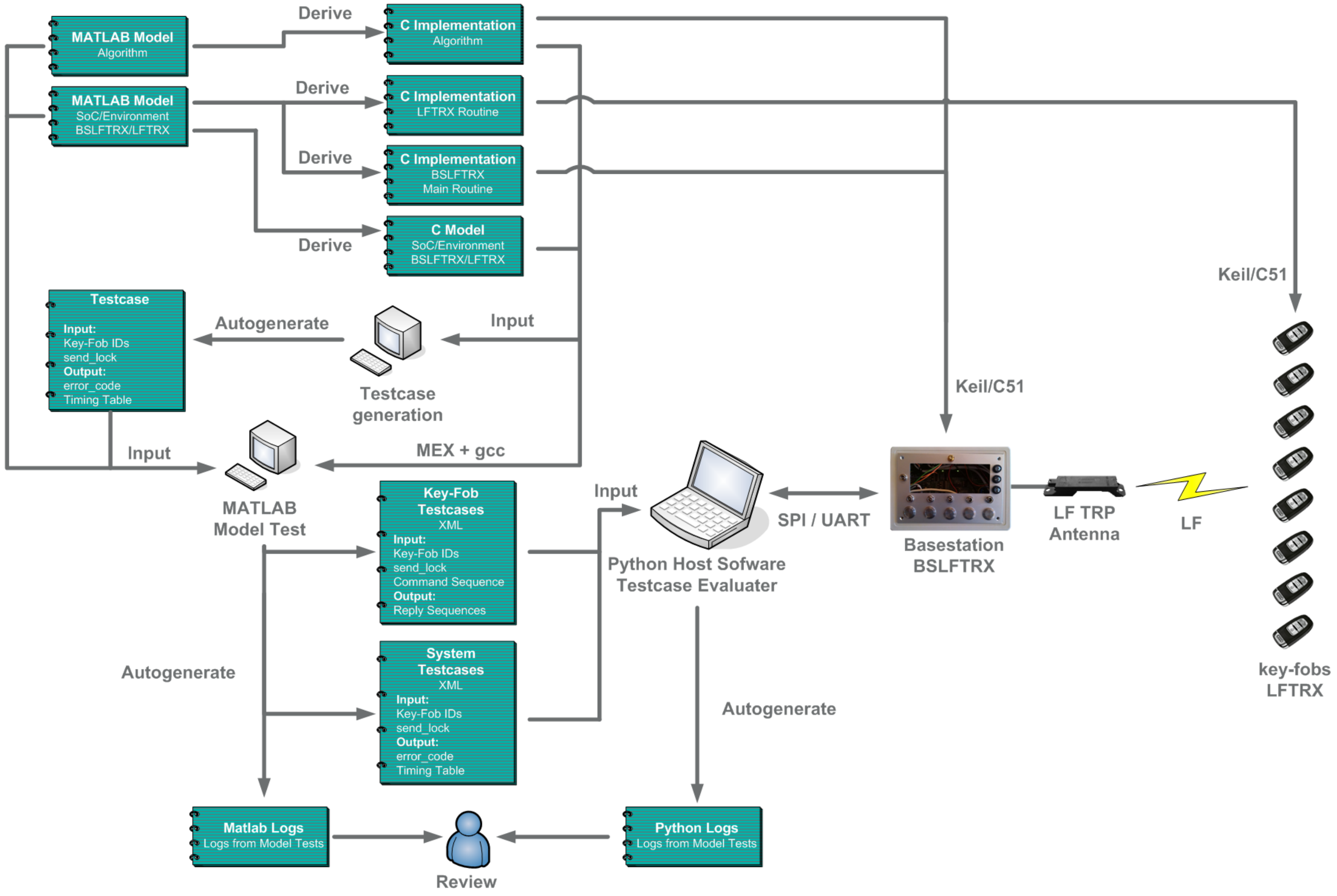

Figure 7.3: Structure of the testframework.

prepared to instrument CREST also on multi-file programs, like the implementation of the pairing procedure with collision management. In order to reuse the generated test-cases, CREST is able to export the input-vectors to text-files.

To use CREST on SW, the tested code has to be preprocessed. This means, that the source code has to compile using a common gcc compiler, and symbolic variables have to be defined. For further information see section C.2.1 in the appendix.

mex and *gcc*

The MATLAB *mex* framework is used together with *gcc* to compile the implementation of the algorithm (in C), so it can be included in a MATLAB framework as MEX file. Therefore all SoC specific defines and all commands, using SoC specific functions, need to be replaced by a C model. Figure 7.4 illustrates the structure of the algorithm, which is compiled using MEX and gcc to be included in MATLAB, whereas all replaced function blocks are colored in red.

MATLAB Model Test Framework

The MATLAB Model Framework includes the *mex* compiled C implementation and the MATLAB Model of the algorithm and applies all generated test-cases on both, to check for conformance. Furthermore a log-file containing a test-statistic as well as XML-test-case definition files for the HW-in-the-Loop tests are generated.

7.4.2 Hardware-in-the-Loop

Python Host SW and Test-Case-Evaluator

The Python Host SW reads the XML-test-case definition files and configures the HW setup to perform the same test-cases already done for the SW-in-the-Loop tests. It communicates with the BS via 3WI or UART and can also have the BS forward debug and configuration commands to the Key-Fobs. The XML-test-case definition files contain the desired reply of the BS for each test-case execution. Using this information, the Python Host also acts as test-case Evaluator and generates logs and test statistics.

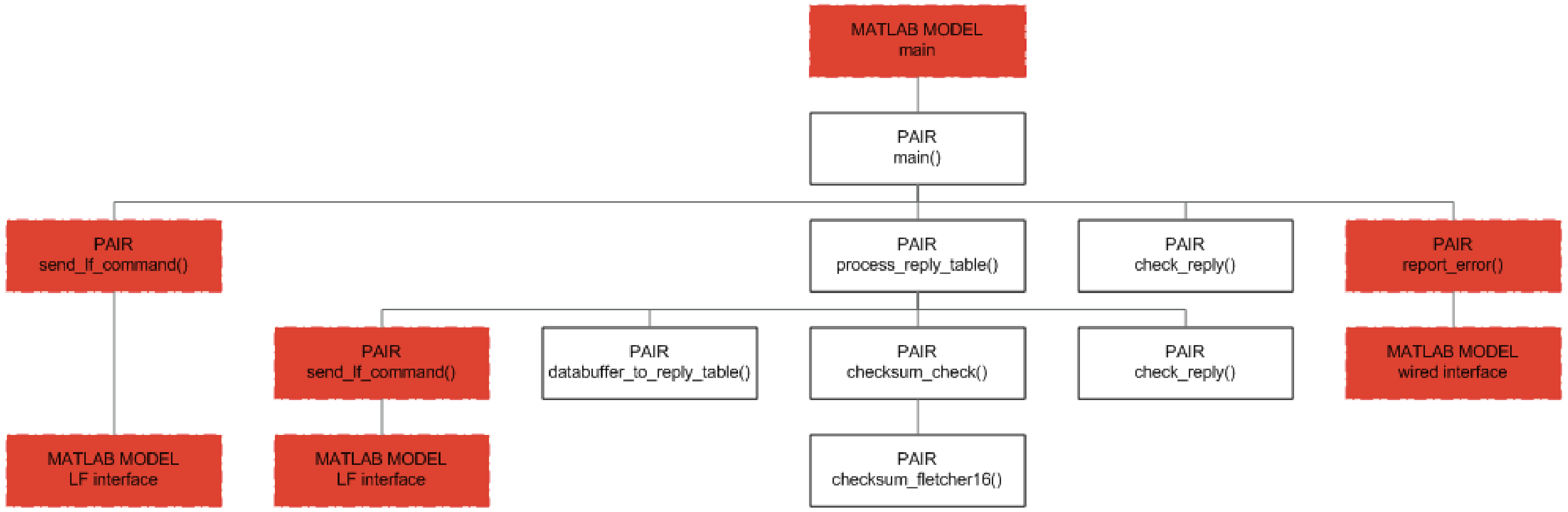

Figure 7.4: FW structure of the algorithm implementation which is compiled using *mex* and *gcc*. The colored blocks are model functions, replacing blocks using SoC specific functions.

HW Setup

The HW setup consists of one BS with LF TRP antenna and eight Key-Fobs. The communication between BS and computer can be established either via 3WI or via UART. As link between the Key-Fobs and the BS the LF-TRP channel is used.

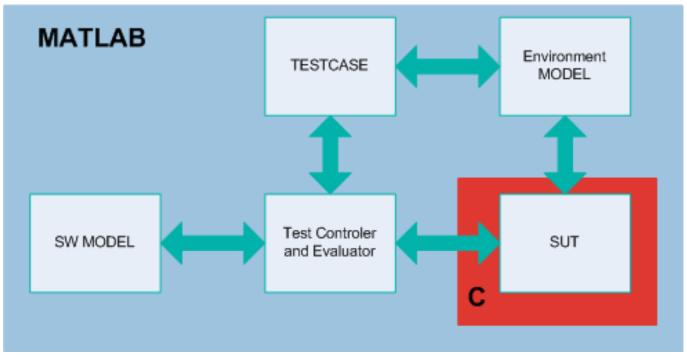

Figure 7.5: Structure of the MATLAB Model Testing Framework.

7.4.3 Manual Review

In order to prevent false evaluation of HW-in-the-Loop tests due to HW errors, a manual review of the log-files is advised to get a final overview over the test-process and performance of the system.

7.5 Key-Fob Tests

7.5.1 Code Analysis

The code analysis shows that the LFTRX SoC (Key-Fob) FW consists of 651 Lines Of Code (LOC), which are mainly SoC specific statements. Also the code is quite trivial, as the Key-Fob is just event triggered and does not implement any algorithm-intelligence. Therefore, a dynamic SW in the loop testing not feasible and this part of the FW will just be analysed statically and in the HW loop. For those HW-in-the-Loop tests manually defined test-cases and test-cases generated by the MATLAB Model Test Framework during the Model Tests of the BS FW are used.

7.5.2 Software-in-the-Loop

Static code analysis with *LINT* and Motor Industry Software Reliability Association (MISRA) 2004 rules was performed and terminated with zero errors, warnings and messages. For further information concerning *LINT* with MISRA see [Gim01].

7.5.3 Hardware-in-the-Loop

For the HW-in-the-Loop tests the XML-test-case specification files are used, whereas one test-case consists of a series of LF commands send to the Key-Fobs. The BSLFTRX SoC does not implement any functionality and just works as bridge between the wired interface and the LF link from the Computer to the Key-Fobs. Each command send to the Key-Fobs can be evaluated separately and a test-case is considered successful if all commands sent to the Key-Fob result in the desired replies.

As collisions cannot be specified in the XML files accurately, commands forcing collisions result in fail-evaluation of the test-case. Therefore a manual evaluation of the log-files is necessary. The flow of the test can be seen in figure 7.6 (left).

79 test-sequences generated from the model tests and 28 manually defined test-cases were used for the Key-Fob HW-in-the-Loop tests. Those testcases are defined for different setups with varying number of Key-Fobs present in range of the TRP antenna. The tests showed that there is a bug in the LFTRX SoC when sending data in the 8^{th} timeslot, as all testcases in which Key-Fobs were supposed to send data in timeslot 8, failed. This bug was investigated further with manual tests and later-on fixed with a workaround in the FW, to prevent usage of the defective SoC functionality. The SoC bug has to be further investigated by the Design Team, but does not affect the functionality of the FW for now.

Furthermore it showed, that the LF link is quite unstable. The error in the LF transmission did not occur deterministic throughout the test sequences and could not be identified as an FW error, neither on the Key-Fob nor on the BSLFTRX. This means that the instability may be caused by imperfect measurement and laboratory Printed Circuit Board (PCB), which are used in the development progress. Further investigation with the help of Application Engineers to verify the functionality of the laboratory PCB is needed, in order to isolate possible error sources.

Expect those HW induced problems, the FW worked fine and was tested on several different setups, where no FW errors could be found.

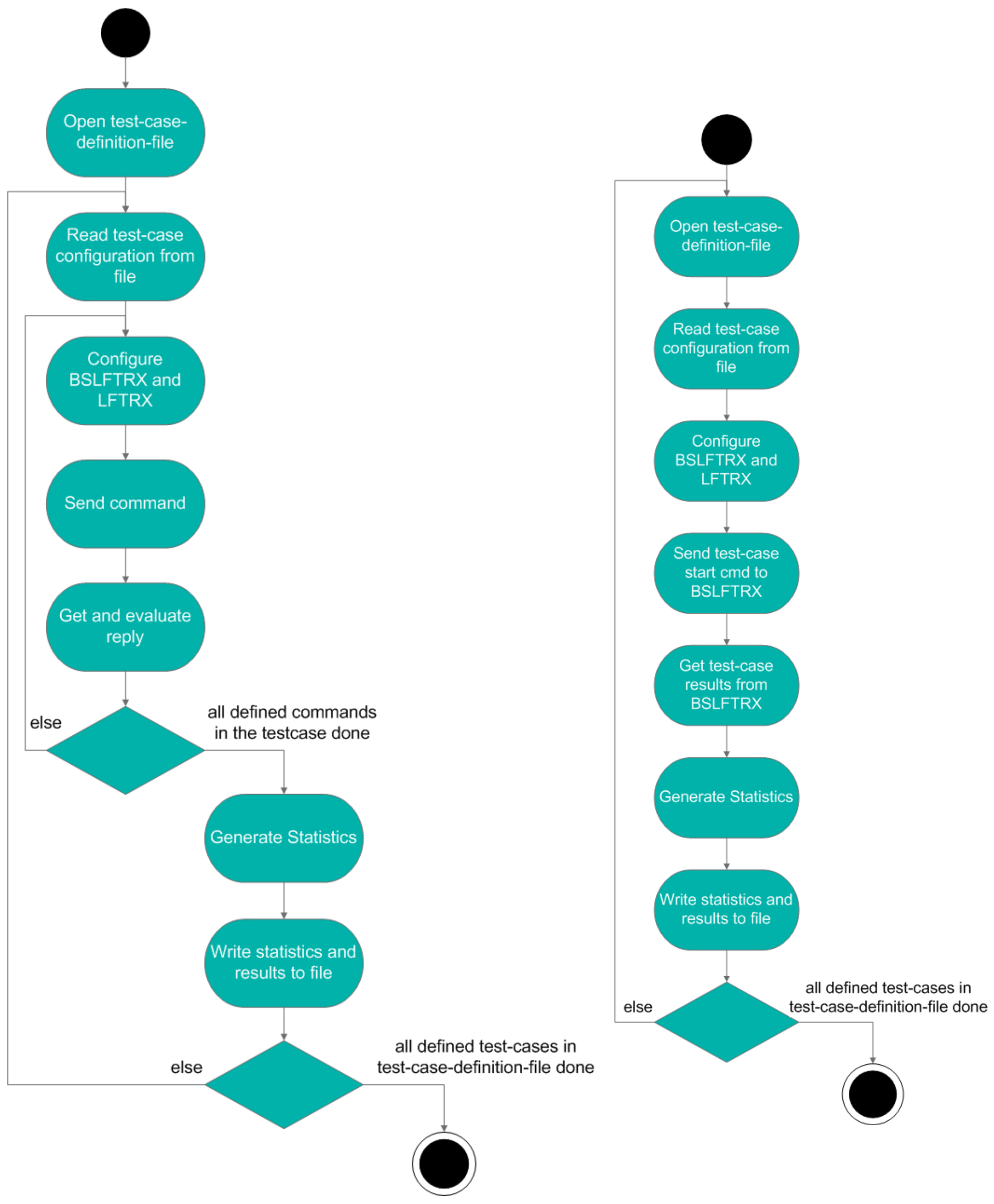
Open test-case-definition-file
Read test-case configuration from file
Configure BSLFTRX and LFTRX
Send command
Get and evaluate reply
else
all defined commands in the testcase done
Generate Statistics
Write statistics and results to file
else
all defined test-cases in test-case-definition-file done
Open test-case-definition-file
Read test-case configuration from file
Configure BSLFTRX and LFTRX
Send test-case start cmd to BSLFTRX
Get test-case results from BSLFTRX
Generate Statistics
Write statistics and results to file
else
all defined test-cases in test-case-definition-file done

Figure 7.6: *Left:* Execution flow of the Key-Fob HW-in-the-Loop tests. *Right:* Execution flow of the BS HW-in-the-Loop tests.

7.6 Base Station Tests

7.6.1 Code Analysis

Analysis of the BSLFTRX SoC FW code shows, that it contains the main part of the algorithm-intelligence, and is way more complex than the FW code of the LFTRX SoC in the Key-Fob. Also the implementation focused on modularity, to separate the algorithm as good as possible from the other parts of the FW. Therefore the algorithm uses mainly standard C functions and statements and has just a few interfaces to the SoC specific functions.

For those reasons a SW-in-the-Loop testing is not just desirable, due to complexity of the code, but also highly feasible, as there is only a small interface to the SoC specific functions which can easily be modeled within the test framework. The pure pairing and collision management procedure has 1532 LOC, which are tested SW-in-the-Loop to verify the procedure. The whole BSLFTRX FW, which has 2378 LOC, is just tested HW-in-the-Loop, as it contains also a lot of SoC specific commands.

7.6.2 Software-in-the-Loop

Static code analysis with *LINT* and MISRA 2004 rules was performed and terminated with zero errors, warnings and messages. For further information concerning *LINT* with MISRA see [Gim01].

As second step, test-case generation with CREST (see [BS08] and section 7.4.1) was done. Using Control Flow Directed Search, CREST was able to evaluate 164 of 172 branches. The remaining branches could not be evaluated by CREST, as those branches are catching errors which can just occur on malfunction of the HW and could not be modeled. Those errors are forced to occur in the HW-in-the-loop tests to also cover this branches of the algorithm. In order to have more test-cases which can later-on be used in the Matlab Model Test Framework and in the HW-in-the-loop tests, CREST was instructed to run 50 iterations generating random input data and export the generated input to files. In addition, to cover extreme cases and special error cases, also 29 manual test-cases where written, which could later-on be used in the Model Testing Framework.

Third, dynamic code testing was done, in form of Black Box Tests performed with MATLAB. Therefore a framework to test all single modules (see figure 7.2) of the algorithm as well as the whole implementation against a MATLAB model was implemented using MATLAB *mex*. As input for the Black Box Tests, the 50 testcases generated by CREST, as well as 29 manually defined testcases were used. With those tests, it was possible to verify the functionality of the procedure and all its modules, and find some common programming errors, as typing or off-by-one errors, in the code. Furthermore an overflow error in the Fletcher-16 calculation, which had different effects in the MATLAB model (double-precision arithmetic) and the implementation (8-bit precision arithmetic), was discovered.

7.6.3 Hardware-in-the-Loop

Using the generated XML-test-case specification files a series of HW-in-the-Loop tests were performed. For that the 79 test-cases where used, which were previously converted to XML testcase specification files. The flow of the test can be seen in figure 7.6 (right). It showed that the algorithm is highly reliable and many occurring HW errors can be compensated. As already observed in the Key-Fob tests (section 7.5.3), the LF link was unstable, which is not caused by the FW. As the algorithm was already tested and corrected during the SW-in-the-Loop tests, and the rest of the BSLFTRX FW consisted mainly of function calls to tested and verified SoC functions provided by Maxim Integrated, there where no further FW errors that could be identified by these tests.

Chapter 8

Conclusion and Outlook

8.1 Evaluation

The evaluation of the KGO System and the state-of-the-art procedures for pairing and collision management showed, that it is not possible to implement a state-of-the-art procedure from literature without modification. This is due to the special characteristics and capabilities of the components of the KGO System. In order to use those characteristics effectively, a procedure, tailored to the KGO System, had to be designed. Also, as the pairing is designed for a limited operation mode as it should be possible to pair a Key-Fob without battery, it showed that no proper random number source is usable on the Key-Fob. Therefore a deterministic procedure had to be chosen for implementation. This could have lead to problems in systems with unlimited number of Key-Fobs, but as the number is limited to eight, many error cases and could be explored and taken care of when implementing, to make the procedure work properly for the KGO System setup.

The tests show that the implementation is working fine and fulfills all the specification, defined at the project start, which are:

- Initial communication with up to eight Key-Fobs.
- Slot assignment procedure.
- Dummy Transmission which can later-on replaced with Key-Fob-BS communication for car-credentials, cryptographic key and additional necessary data.
- Verifying of the pairing success, including a simple error handling.

Furthermore the algorithm is very robust and can easily be extended to be less error prone. Also the design is easily portable, so it can easily be integrated in other FW projects or demonstrators of the KGO System.

Moreover the test framework can be considered a useful output of the project. It is tailored to the needs of the KGO System, but can easily be reused as base for a more powerful and generic functional testing framework for SoC FW. It connects SW-in-the-Loop and HW-in-the-Loop tests to allow an easy test and evaluation of the performance of the tested FW, because its easy to isolate the tests for specific components of the system. This is due to the fact that all components can be tested separately, SW as well as Key-Fob-FW and BS-FW implementation on the HW. Also the biggest part of the tests are automated, the manual effort is minimized to setting up the framework and evaluating the results. Although it needs to be mentioned, that the quality of the tests highly depend on the quality of the models and the test-case-generator of the framework.

8.2 Future Work

For further development of the procedure, additional security features, e.g. cryptography, to make the procedure safe against cryptographic attacks should be implemented. This would also enable that the pairing procedure could be triggered also by the carholder, not just the OEM or a mechanic. In case one Key-Fob stops working or gets lost or a new Key-Fob needs to be added to the car on a customer request, a single pair, or single un-pair of Key-Fobs also needs to be added.

At the moment, the procedure also just implements a very rudimentary error handling, which mainly consists of error reports. In order to make it more independent and automated, a more sophisticated error detection and error handling could be implemented, also considering changes in the setup during the process or failures in the Key-Fobs. A good error detection and error handling would also increase the procedures security and decrease its proneness to attacks.

In addition, also a final verification of the pairing procedure could be implemented, for example using a challenge-response-procedure. This would further increase the reliability of the system and the susceptibility to errors. Moreover the testing framework should also be extended by a more sophisticated test-case generator, in order to have a better test coverage (Path Coverage or Modified Condition/Decision Coverage).

Appendices

Appendix A

Characterization of the Random Number Sources available on the Key-Fob in TRP Mode

A.1 Measurement Description

For the stochastic collision management process, a sufficient random number source would be necessary to find the reply timeslot for the Intelligent Key Unit used in the Keyless Go System (Key-Fob). For this reason all possible random number sources available on the Key-Fob LF Transceiver SoC (LFTRX) System-on-Chip (SoC) on the Key-Fob had to be identified and characterized. As during Transponder (TRP) mode just particular modules of the Key-Fob are active, not all possible sources could have been taken into consideration. Possible sources are:

- VDDI - the inductive supply voltage of the LFTRX SoC on the Key-Fob. This voltage depends on the strength of the Low Frequency (LF) field.
- VTP - temperature voltage, depending on the environmental temperature

Measuring the Received Signal Strength Indication (RSSI) is not an option, as the LFTRX SoC does not offer RSSI measurement in TRP mode, as the analog front end to the RSSI pins is not active.

Furthermore, this measurement setup was used to determine if the dominant sender problem can occur in this system. This would cause some more features in the implementation of the final algorithm.

A.1.1 Measurement Setup

On the Computer a python host Software (SW) is running to control the measurement flow. It communicates with the Base Station (BS) via 3-Wire-Interface (3WI) or Universal Asynchronous Receiver/Transmitter (UART), using a Maxim Integrated FDTI interface device. The BS executes the control commands send by the computer and forwards measurement commands to the Key-Fob, using the LF TRP communication channel. The Key-Fob receives the command and executes the measurement, whereas per command three samples are taken. Those three samples are then send to the BS via LF TRP communication and from there forwarded to the Computer via 3WI or UART where they are processed and stored by the host SW.

For the measurements of the random number sources just one Key-Fob is placed in range of the LF TRP signal, to make sure no collisions occur, whereas the Key-Fobs vary. On the other hand, for the dominant sender problem, eight Key-Fobs will be placed in range to provoke collisions and keys sending at the same time.

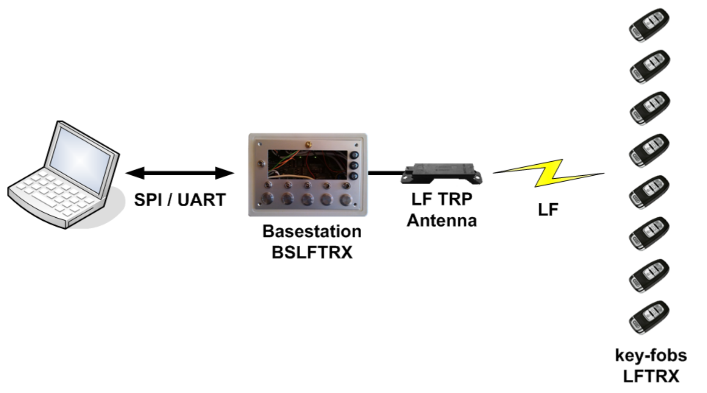

Figure A.1: Measurement setup for doing the measurements on the Key-Fob and identifying the dominant sender problem.

The Key-Fobs were either placed on a rooster in a horizontal plane or in a cup vertically aligned, to simulate various placement methods within a car during production. Furthermore different Key-Fobs where used for every position and measurement type (see figure A.2). The LF TRP coil was fixed in the inside of a box beneath the rooster.

A.2 Measurement Results

The measurements have been done for 10 different Key-Fobs, at 6 different position in horizontal (for a flat surface) and 2 in vertical (for a cup-holder). There were 300 samples

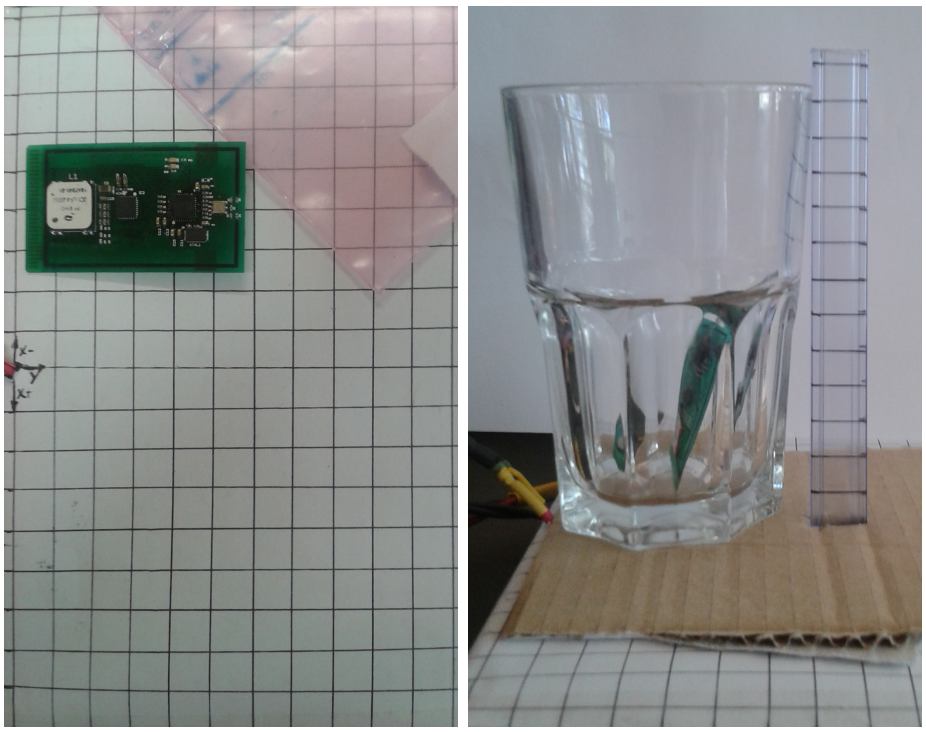

Figure A.2: Measurement setup for characterizing the random number sources on the Key-Fob depending on the position of the Key-Fob simulating a cup-holder and a flat space near the LF TRP coil.

taken per Key-Fob and position with a Analog to Digital Converter (ADC) resolution of 10 bit.

VDDI

Figure A.3 shows that the probability distribution is too narrow for a cup-holder formed as well as for a flat surface above the LF TRP antenna. Therefore its not sufficient to use VDDI as random number source for selecting the reply timeslot, as a lot of collisions would occur.

This is because the distance between the Key-Fob and the LF TRP antenna is small enough, so the Key-Fob can load the capacitor, used for storing energy from the LF field, high enough that VDDI is at the limit[1].

[1]A limiter is controlling the maximum of the VDDI, to prevent VDDI from getting bigger than the allowed maximum if the LF TRP field is very strong.

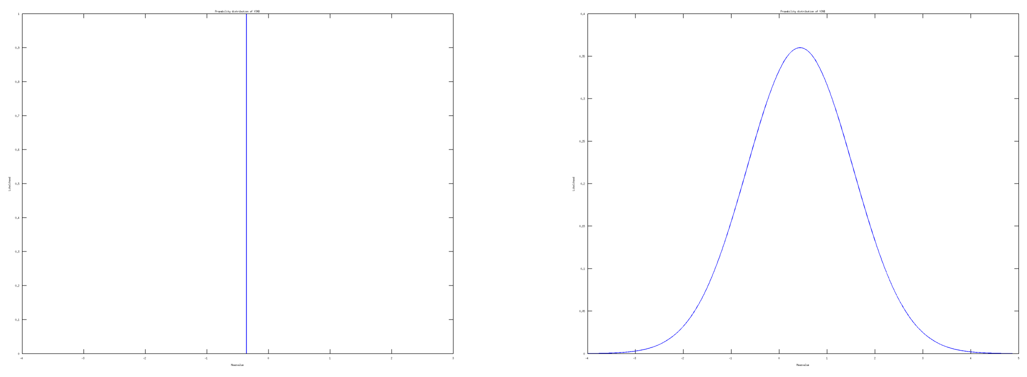

Figure A.3: Probability distribution of the measurement values taken at different positions for VDDI. (left - flat space; right - cup-holder).

VTP

As in practice the temperature in a normal environment will not show big changes, therefore measurements in normal laboratory environment without any temperature control is sufficient.

Measurements show that the distribution depending on the position of the Key-Fob for a cup-holder as well as for a flat surface this behavior is not sufficient (see figure A.4).

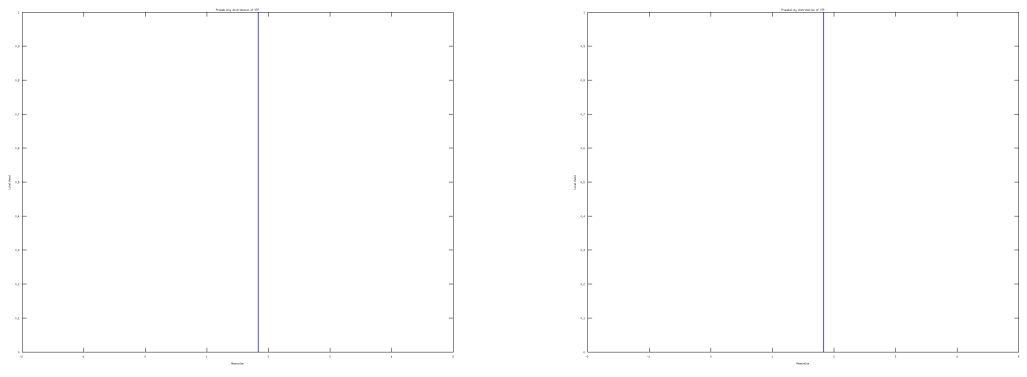

Figure A.4: Probability distribution of the measurement values taken at different positions for VTP. (left - flat space; right - cup-holder).

Dominant Sender Problem

To identify if the dominant sender problem occurs in this setup, multiple measurement where taken, whereas for one measurement just one Key-Fob was in range of the LF TRP antenna. For the other measurements, eight Key-Fobs where randomly arranged in the field. For each measurement 1000 packets were send.

As figure A.5 shows, that for one Key-Fob in range, all packets could be received without any problems. For eight Key-Fobs just 20% of the packets could be received, less than 10% were corrupt and in more than 70% of the cases no valid packet could be received.

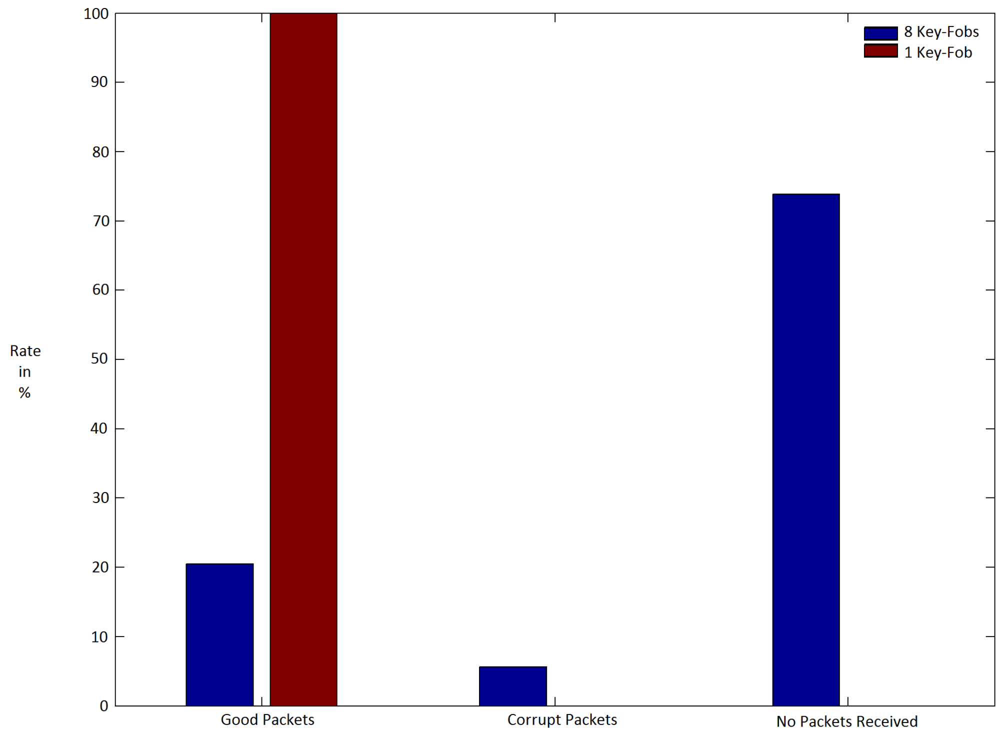

Figure A.5: Measurement setup for doing the measurements on the Key-Fob and identifying the dominant sender problem.

A.3 Conclusion

The measurements show, that no available random number source on the Key-Fob is reliably and therefore useable in practice. This means an algorithm relying on a good random number source is not an option for the implementation. Furthermore it is shown that the dominant sender can occur in some few cases, and in order to build a reliable system, this must be considered in the design of the algorithms.

Appendix B

Definition of Conformance

The observations made in the following sections are based on the observations and assumptions made in [Tre96]

B.1 Symbol Description

B.1.1 Model

The model m describes the algorithm and is verified to be correct.

B.1.2 Implementation

The Implementation i is derived from the model and represents the Implementation Under Test (IUT).

B.1.3 Input

A set of inputs x consists of the 8x3 matrix; one (Identification number (ID), SEND_LOCK, timeslot number) triple for each Intelligent Key Unit used in the Keyless Go System (Key-Fob).

The input set x can be chosen from all possible inputs in X, whereas a valid input set relies following restrictions:

i) All IDs in one set have to be unique.

ii) SEND_LOCK $\in \{0, 1\}$

iii) timeslot number $\in [0, 7]$

B.1.4 Output

A output set consists of the tuple (error_code, timing table). The error_code is a scalar and timing table a $8x2$ matrix, as defined in section 5.4.2.

B.1.5 Equality

Software-in-the-Loop

The equality operator $==_{out\_SW}$ for two output sets $o_1 = (\text{error\_code}_1, \text{timing table}_1)$ and $o_2 = (\text{error\_code}_2, \text{timing table}_2)$ is defined as follows:

$$
\begin{aligned}
& o_1 == o_2 \textbf{ iff} \\
& \quad \text{error\_code}_1 == \text{error\_code}_2 \\
& \quad \forall idx \in [0, 7] : \text{timing table}_1[idx].valid == \text{timing table}_2[idx].valid \\
& \quad \forall idx \in [0, 7] : \text{timing table}_1[idx].key\_fob\_id == \text{timing table}_2[idx].key\_fob\_id
\end{aligned}
$$

Hardware-in-the-Loop

The equality operator $==_{out\_HW}$ for two output sets $o_1 = (\text{error\_code}_1, \text{timing table}_1)$ and $o_2 = (\text{error\_code}_2, \text{timing table}_2)$ is defined as follows:

$$
\begin{aligned}
& o_1 == o_2 \textbf{ iff} \\
& \quad \text{error\_code}_1 == \text{error\_code}_2 \\
& \quad \forall idx \in [0, 7] : \text{timing table}_1[idx].key\_fob\_id \in \text{timing table}_2 : \\
& \quad idx_2 \in [0, 7] : \text{timing table}_1[idx].key\_fob\_id == \text{timing table}[idx_2]_2.key\_fob\_id \\
& \quad idx_2 \in [0, 7] : \text{timing table}_1[idx].valid == \text{timing table}[idx_2]_2.valid
\end{aligned}
$$

B.2 Derivation of conformance

A implementation of the algorithm is conform to a model if the output for model and implementation equal for the same input set.

$$
\begin{aligned}
i \ \ldots \ \text{Implementation} \qquad & x \ldots \ \text{Input Set} \\
m \ \ldots \ \text{Model} \qquad & X \ldots \ \text{Possible Input Sets}
\end{aligned}
$$

$$i \leq_{conf} m =_{def} \forall x \in X : \textbf{out}\,(i\ \textbf{for}\ x) ==_{out} \textbf{out}\,(m\ \textbf{for}\ x) \tag{B.1}$$

whereas the $==_{out}$ operator can either be replaced by $==_{out\_SW}$ or $==_{out\_HW}$.

Appendix C

Setting up the Test Framework

C.1 Overall Description

Figure C.1 illustrates the structure of the Maxim Integrated Keyless Go System (KGO System) test/development framework. To follow the data-flow start at the upper left corner. The automatic test-case generator, in this case CREST ([BS08]), is used to generate input-sets to test the C implementation.

C.2 Setting up the Components

C.2.1 CREST

To setup CREST for a UNIX system, the following packets need to be installed

- build-essentials
- gcc
- glibc
- yices version 1.0.39 (http://yices.csl.sri.com/download-old.shtml)
- CREST https://github.com/jburnim/crest

yices and CREST can be installed by downloading the archive from the website and following the instructions in the README.

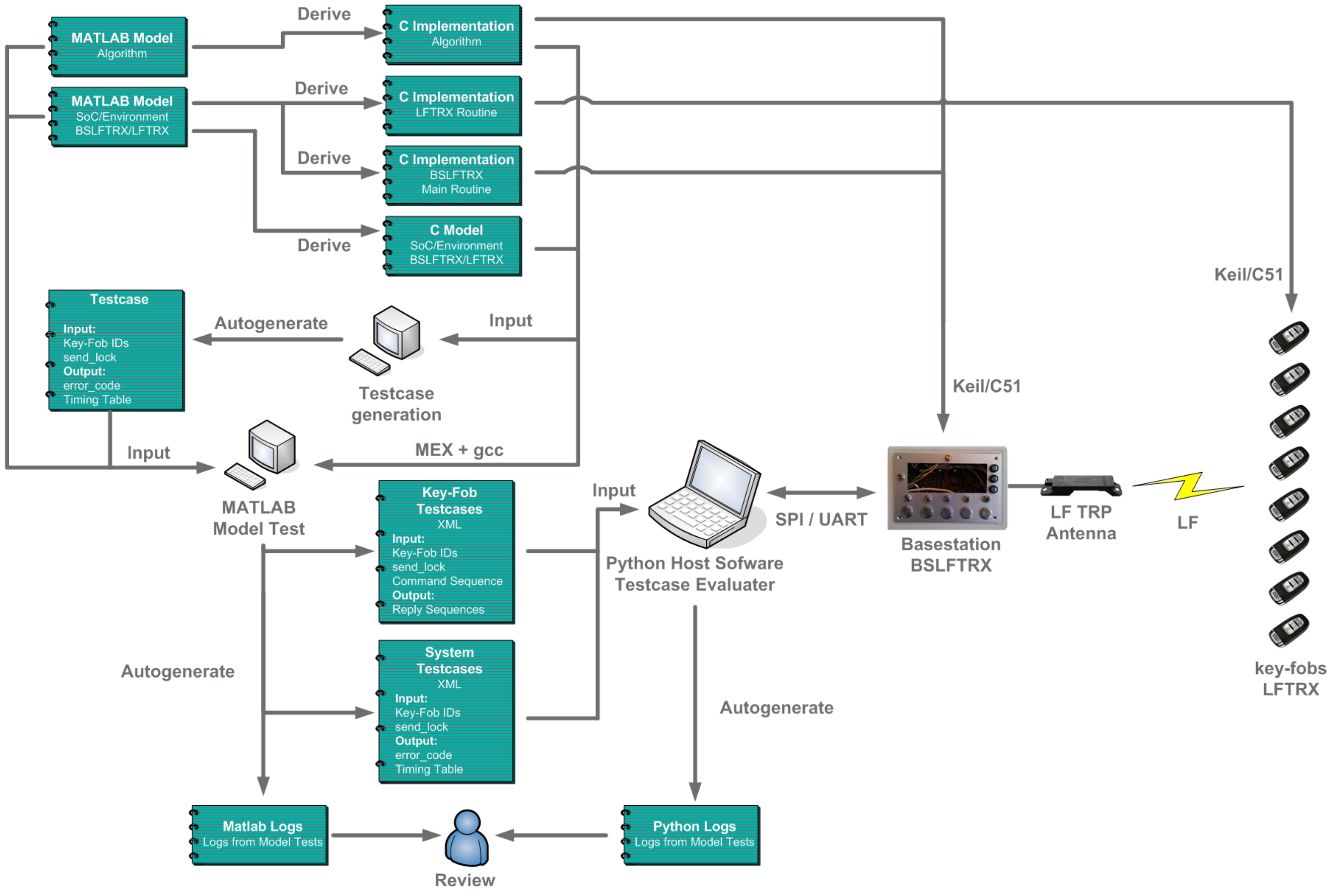

Figure C.1: Structure of the testframework.

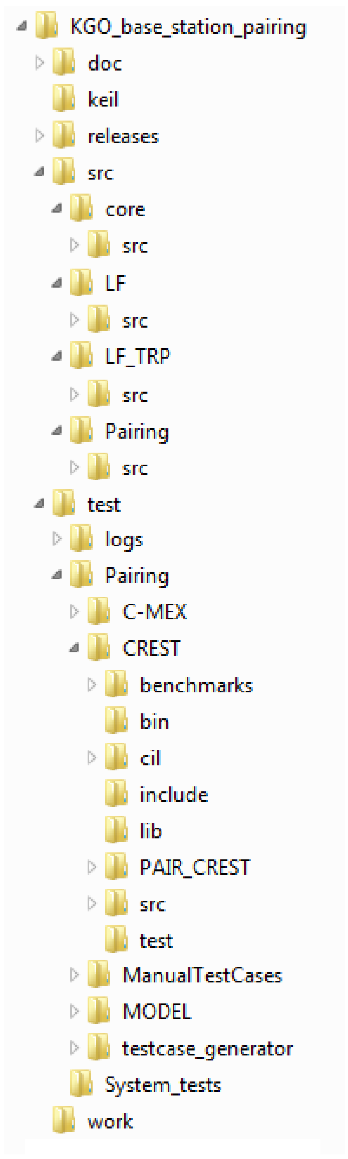

Figure C.2: Folder-structure.

Preparing the algorithm

In order to instrument CREST to generate test-cases, first a dummy main has to implemented, you can see the listing with comments below:

```
uint32 testcase_key_fob_id[LF_TRP_NUM_TIMESLOTS];
uint8 testcase_key_fob_tsn[LF_TRP_NUM_TIMESLOTS];
uint8 testcase_send_lock[LF_TRP_NUM_TIMESLOTS];

void main()
```

```
{
    uint8 cnt = 0;

    static timing_entry xdata timing_table[LF_TRP_NUM_TIMESLOTS];

    uint32 testcase_key_fob_id[LF_TRP_NUM_TIMESLOTS];
    uint8 testcase_key_fob_tsn[LF_TRP_NUM_TIMESLOTS];
    uint8 testcase_send_lock[LF_TRP_NUM_TIMESLOTS];

    CREST_int( testcase_key_fob_id[0]  );
    CREST_int( testcase_key_fob_id[1]  );
    CREST_int( testcase_key_fob_id[2]  );
    CREST_int( testcase_key_fob_id[3]  );
    CREST_int( testcase_key_fob_id[4]  );
    CREST_int( testcase_key_fob_id[5]  );
    CREST_int( testcase_key_fob_id[6]  );
    CREST_int( testcase_key_fob_id[7]  );

    for(cnt = 0; cnt < LF_TRP_NUM_TIMESLOTS; cnt++)
    {
        testcase_key_fob_tsn[LF_TRP_NUM_TIMESLOTS] = 0;
        testcase_send_lock[LF_TRP_NUM_TIMESLOTS] = 0;
    }

    uint8 error_code = PAIR_main(timing_table);
}
```

Second, all algorithm source code needs to be copied into one file. This is done with the *Prepare_C_file.sh* script, in the ./test/Pairing/CREST/PAIR_CREST folder. The resulting file then has to be manually checked as the script may produce non-compile-able code.

C.2.2 Matlab Model Testing Framework

The Framework consists of several components:

- Core script, which handles the execution flow of the framework as well as the test-case evaluation and generating of the log files and compiling of the C implementation to *mex*.
- TESTCASEGEN files are part of the Extensible Markup Language (XML)-File generator. These functions are used to generate the needed test-case specifications on-the-fly, during the Software (SW)-in-the-Loop tests.
- MODEL_PAIR functions implement the Matlab model of the pairing algorithm with

collision management.

- The directories CREST and Manual-test-cases hold the input vector files for the test-cases.
- C-MEX holds the C environmental model file as well as all compiled *mex* files.

No further configuration is needed if all scripts are executed properly. Paths to new test-case input vector files can be added in *TEST_framework_main.m*.

C.2.3 Hardware Setup

The Hardware (HW) setup consists of a Base Station (BS) (Base Station LF Transceiver SoC (BSLFTRX)) and at least nine Key-Fobs (Key-Fob LF Transceiver SoC (LFTRX)), whereas the number of Key-Fobs has to be manually changed for the different test-cases. The BSLFTRX has to be programmed with the *KGO_base_station_pairing* Firmware (FW) and all used Key-Fobs with *KGO_key_pairing_fw*. It has to be noted that it is necessary to set the right debug ID and debug timeslot for the Key-Fobs in the *KGO_key_pairing_fw* FW, before compiling and programming, in order to be able to automatize the test-flow.

DBG_ID	DBG_TSN
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	0

Table C.1: Debug values for the Key-Fobs.

C.3 Testing

LINT is included in the Keil framework provided by Maxim Integrated, using the MISRA 2004 rules. In order to run crest following commands have to be executed:

```
cd PAIR_CREST
../bin/crestc all\_c\_fct.c
../bin/run_crest ./all\_c\_fct 50 -random_input
```

Then TEST_framework_main.m in ./test/Pairing has to be executed to run the Matlab Model Testing Framework. This framework will generate the test-case XML files in ./test/Pairing.

For the hardware setup, the script provided in ./test/System_tests need to be executed, whereas first the absolute paths in the script need to be adjusted. This will generate log-files for all the executed test-cases, which can be found in the same directory. For all test-cases the right number of Key-Fobs in the HW setup have to be set manually.

Appendix D

Packet Definition

D.1 REQUEST

This command requests a specific group of Key-Fobs to reply to the Base Station (BS) with their ID.

D.1.1 Base Station to Key-Fob

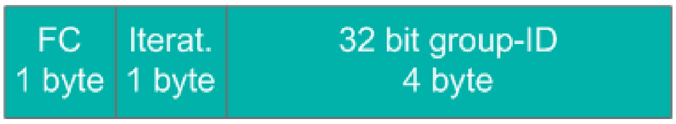

Figure D.1: REQUEST-Packet dataload downlink.

D.1.2 Key-Fob to Base Station

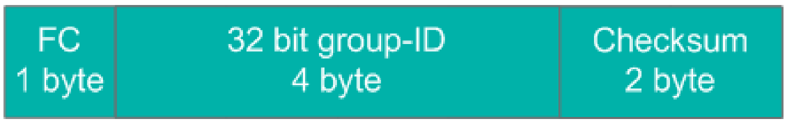

Figure D.2: REQUEST-Packet dataload downlink.

D.2 SET_TIMING

Sets the permanent timeslot number for a specific Key-Fob.

D.2.1 Base Station to Key-Fob

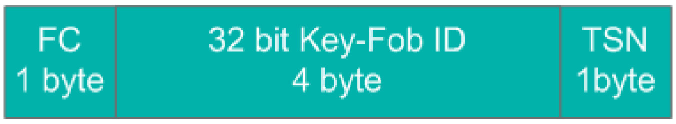

Figure D.3: SET_TIMING-Packet dataload downlink.

D.2.2 Key-Fob to Base Station

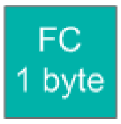

Figure D.4: SET_TIMING-Packet dataload downlink.

D.3 START_CRED_EXCH

Starts the credential exchange between BS and Key-Fob to finalize the pairing procedure.

D.3.1 Base Station to Key-Fob

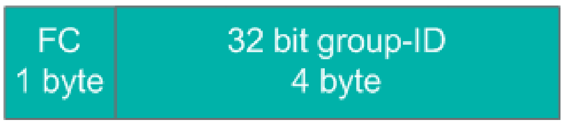

Figure D.5: START_CRED_EXCH-Packet dataload downlink.

D.3.2 Key-Fob to Base Station

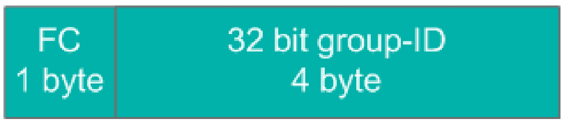

Figure D.6: START_CRED_EXCH-Packet dataload downlink.

D.4 CLEAR

Resets the Key-Fob and forces it to perform a page erase.

D.4.1 Base Station to Key-Fob

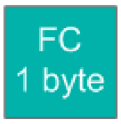

Figure D.7: CLEAR-Packet dataload downlink.

D.4.2 Key-Fob to Base Station

No response by the Intelligent Key Unit used in the Keyless Go System (Key-Fob).

D.5 DBG_SET_32_BIT_ID

This command is for testing and debugging purposes. A temporary 32 bit ID for the LFTRX SoC on the Key-Fob can be set.

D.5.1 Base Station to Key-Fob

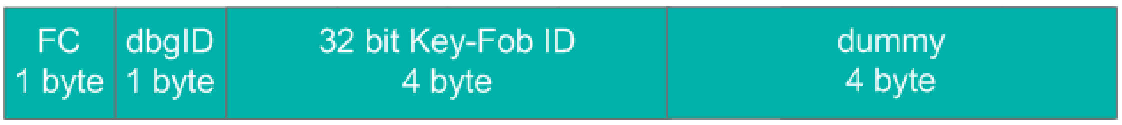

Figure D.8: DBG_SET_32_BIT_ID-Packet dataload downlink.

The dummy bytes are necessary to ensure that the Key-Fob has enough time to process the command.

D.5.2 Key-Fob to Base Station

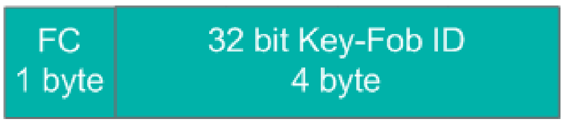

Figure D.9: DBG_SET_32_BIT_ID-Packet dataload downlink.

Appendix E

Abbreviations

3WI	3-Wire-Interface
AES	Advanced Encryption Standard
AMBA	Advanced Microcontroller Bus Architecture
APB	AMBA Peripheral Bus
API	Application Programming Interface
ACK	Acknowledge
ADC	Analog to Digital Converter
ASK	Amplitude Shift Keying
BIST	Build-In-Self-Test
BPSK	Binary Phase Shift Keying
BS	Base Station
BSLFTRX	Base Station LF Transceiver SoC
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
CSMA/CD	Collision Sense Multiple Access with Collision Detection
CSMA/CA	Collision Sense Multiple Access with Collision Avoidance
CTS	Clear-to-Send
CRC	Cycle Redundancy Check

DC	Direct Current
DSP	Digital Signal Processing
ECC	Elliptic Curve Cryptography
ESP	Electronic Stability Program
FDMA	Frequency Division Multiple Access
FW	Firmware
FSK	Frequency Shift Keying
GPIO	General Purpose Input Output
HW	Hardware
ID	Identification number
I/O	Input/Output
IUT	Implementation Under Test
ISM-Band	Industrial, Scientific and Medical radio band
JLCC	JTAG Like Control Chain
JTAG	Joint Test Action Group
KGO System	Maxim Integrated Keyless Go System
Key-Fob	Intelligent Key Unit used in the Keyless Go System
LF	Low Frequency
LFTRX	Key-Fob LF Transceiver SoC
LMP	Link Manager Protocol
LOC	Lines Of Code
MISRA	Motor Industry Software Reliability Association
MSK	Minimum Shift Keying
NRZ	No-Return-to-Zero
OEM	Original Equipment Manufacturer
OOB	Out-Of-Bound Channel
PCB	Printed Circuit Board

QPSK Quadrature Shift Keying

RAM Random Access Memory

RCO RC Oscillator

RF Radio Frequency

RFID Radio Frequency Identification

ROM Read-Only Memory

RSSI Received Signal Strength Indication

RX Receive

RTS Ready-to-Send

SALOHA Slotted ALOHA

SDMA Space Division Multiple Access

SoC System-on-Chip

SSP Secure Simple Pairing

SMT Satisfiability Modulo Theories

SW Software

SWI Single Wire Interface

TDMA Time Division Multiple Access

TRP Transponder

TRX Transceiver

TX Transmit

UART Universal Asynchronous Receiver/Transmitter

UHF Ultra High Frequency

UHFTRX UHF Transceiver SoC

XML Extensible Markup Language

List of Figures

List of Tables

Bibliography

[AHI12] Abdallah Y. Alma'aitah, Hossam S. Hassanein, and Mohammad Ibnkahla. Modulation Silencing: Novel RFID Anti-Collision Resolution for Passive Tags. In *IEEE International Conference on RFID (RFID)*, pages 81–88. IEEE, 2012.

[BS08] Jacob Burnim and Koushik Sen. Heuristics for Dynamic Test Generation. *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2008. 65, 66, 73, 89

[CDZY10] Wenqing Chen, Hongzheng Dong, Jianzhong Zhou, and Leibo Yao. A RFID Anti-collision Algorithm Based on Two-time-slotted Binary Search. *IEEE*, 2010.

[Che12] Wen-Tzu Chen. A new RFID Anti-collision algorithms for the EPCglobal UHF Class-1 Generation-2 standard. In *9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pages 811–815. IEEE, 2012.

[CLY$^+$10] Xiaoyun Chen, Guohua Liu, Yukai Yao, Yi Chen, Shengfa Miao, and Youli Su. IRBST:An Improved RFID Anti-Collision Algorithm Based on Regressive-style Binary Search Tree. In *International Forum on Information Technology and Applications*, pages 403–406. IEEE, 2010.

[CS06] Richard Chang and Vitaly Shmatikov. Formal Analysis of Authentication in Bluetooth Device Pairing. *The University of Texas at Austin*, 2006.

[CsJ10] Bai Cheng-sen and Zhu Jiang. Research on an RFID Anti-collision Improved Algorithm Based on Binary Search. In *International Conference on Computer Application and System Modeling (ICCASM 2010)*, pages 430–432. IEEE, 2010.

[dAMH08] Marcelo C. de Azambuja, Cesar A. M. Marcon, and Fabiano P. Hessel. Survey of Standardized ISO 18000-6 RFID Anti-Collision Protocols. In *The Second*

International Conference on Sensor Technologies and Applications, pages 468–473. IEEE, 2008.

[DdM06] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). *Computer Aided Verification*, 4144:81–94, 2006. 66

[Ell11] Ellisys. Secure Simple Pairing Explained. Technical report, Ellisys Bluetooth Expert Notes, 2011. 25

[EPC08] EPCglobal. EPC Radio Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860-960 MHz. Specification, EPCglobal, October 2008. version 1.2.0.

[Fel95] David C. Feldmeier. Fast Software Implementation of Error Detection Codes. *IEEE/JACM TRANSACTIONS ON NETWORKING*, 3(6):640–641, December 1995.

[Fin02] Klaus Finkenzeller. *RFID-Handbuch - Grundlagen und praktische Anwendungen induktiver Funkanalagen, Transponder und kontaktloser Chipkarten.* Hanser Verlag, 3rd edition edition, 2002. 15, 17, 18, 19, 20

[Gim01] Gimpel Software, 3207 Hogarth Lane, Collegeville, PA 19426. *Reference Manual for PC-lint/FlexeLint - A Diagnostic Facility for C and C++*, version 8.00 edition, July 2001. 71, 73

[GKY12] Haosong Gou, Sungryul Kim, and Younghwan Yoo. A Bit Collision Detection Based Query Tree Protocol for Sensor Tags in Logistics Management. *ICOIN*, pages 126–131, 2012.

[Gro08] Bluetooth Special Interest Group. Bluetooth Core Specification Addendum. Specification Addendum 1-4, Bluetooth Special Interest Group, 2008.

[GSV11] Alexander Gallego, Nitesh Saxena, and Jonathan Voris. Playful Security: A Computer Game for Secure Wireless Device Pairing. In *The 16th International Conference on Computer Games*, pages 177–184. IEEE, 2011. 26

[HS13] Tzipora Halevi and Nitesh Saxena. Acoustic Eavesdropping Attacks on Constrained Wireless Device Pairing. *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 8(3):563–577, March 2013. 26

[IEE02] IEEE. 802.15.1. Specification 15.1, IEEE, 2002. 23

[IEE05] IEEE. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications. Specifications, IEEE, 2005. Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks-Specific requirements.

[IG99] IEEE and Bluetooth Special Interest Group. Specification of the Bluetooth System. Specification, Bluetooth Special Interest Group, 1999.

[Inf01] Federal Information. Announcing the ADVANCED ENCRYPTION STANDARD (AES). Processing Standards Publication 197, Federal Information, November 2001.

[Int12] Maxim Integrated. Digital Top Specification. LF Basestation SoC - revision V01.5r, May 2012.

[Int13a] Maxim Integrated. 22 kHz StartUp Receiver and 3D Transponder ASIC. LF Transreceiver and Transponder SoC - revision V17.00, May 2013.

[Int13b] Maxim Integrated. Application Note. LF Basestation SoC - revision V02.40, April 2013.

[KH99] Nobuyoshi Komurot and Hiromasa Habuchi. Proposal of a Spread ALOHA System using the SS-CSC Technique. *IEEE*, 1999. 19

[KS09] Ambarish Karole and Nitesh Saxena. Improving the Robustness of Wireless Device Pairing Using Hyphen-Delimited Numeric Comparison. In *International Conference on Network-Based Information Systems*, pages 273–278. IEEE, 2009. 23

[KsJK$^+$09] Hajeong Kim, Young sub Jang, Ki Young Kim, Hanna Cho, Min A Jung, Bongseog Jang, and Seong Ro Lee. Automatic and Adaptive Slot Allocation Method for Initial Ranging Contention Process. *ISCIT*, pages 1307–1309, 2009.

[KSTU09] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5:734–749, 2009. 23

[LG08] Zheng Li and Georges Gielen. Managing Packet Collisions in Scavenging-based ULP Transmit-only Indoor Localization systems. *ICCS*, pages 133–137, 2008. 27

[Li10] Nan Li. Research on Diffie-Hellman Key Exchange Protocol. In *2nd International Conference on Computer Engineering and Technology*, volume 4, pages 634–637. IEEE, 2010. Information Engineering Teaching and research section The People's Armed Police Force Academy of China Langfang Hebei 065000, China.

[LST13] Thomas Luecker, Gerhard Schultes, and Edwin Taferner. Keyless Go Demonstrator Documentation. Maxim Integrated - revision V03.06, April 2013.

[LZ94] Zhao Liu and Magda El Zarki. PERFORMANCE ANALYSIS OF DS-CDMA WITH SLOTTED ALOHA RANDOM ACCESS FOR PACKET PCNS. *PIMRC '94 / WCN*, C 7.2, 1994. 19

[MK09] Theresa C. Maxino and Philip J. Koopman. The Effectiveness of Checksums for Embedded Control Networks. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 6(1):59–72, January-March 2009. 28

[Ngu05] Gam D. Nguyen. Error-Detection Codes: Algorithms and Fast Implementation. *IEEE TRANSACTIONS ON COMPUTERS*, 54(1):1–11, January 2005. 28

[NMRW02] George Necula, Scott McPeak, Shree Rahul, and Westley Weimer. CIL: Intermediate language and tools for analysis and transformation of C programs. *Proceedings of Conference on Compiler Construction*, 2002. 66

[ON09] Takashi Oshiba and Hideaki Nebayashi. Device Pairing Based on Adaptive Channel Fluctuation Control for Large-scale Organizations. *IEEE*, pages 901–906, 2009. Service Platforms Research Laboratories, NEC Corporation, Japan.

[RAD12] Talha Rasheed, Mohamed H. Ahmed, and Octavia A. Dobre. User Pairing for Capacity Maximization in Cooperative Wireless Network Coding. *IEEE*, 2012.

[Sch06] Gerhard Schultes. Draft Circuit Specifications - Intelligent RF transceiver with for battery-less operation with ultra low power management. Maxim Integrated - RF Transreceiver SoC - revision V22.00, Ocotber 2006.

[SEA14] SEAT. Technische Daten und Abmessungen SEAT Alhambra. Figure, SEAT (Sociedad Española de Automóviles de Turismo, S.A), 2014. `http://www.seat.at/alhambra/technische-daten/abmessungen`. 8, 103

[SEKA11] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure Device Pairing Based on a Visual Channel: Design and Usability Study. *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 6(1):28–38, March 2011. 26

[SGJ13] Gerhard Schultes, Oliver Gerler, and Bernd Janger. Circuit Specifications - LF Base Station Antenna Driver. Maxim Integrated - LF Basestation SoC - revision V03.40, June 2013.

[SL03] Dongxu Shen and Victor O.K. Li. Performance Analysis for A Stabilized Multi-channel Slotted ALOHA Algorithm. In *The 14th IEEE 2003 Inter-*

national Symposium on Persona1,lndoor and Mobile Radio Communicafion Proceedings, pages 249–253. IEEE, 2003. 19

[Sta09] Ernst Stadlober. Wahrscheinlichkeitstheorie für informatikstudien. Lecture materials 3, Institut fuer Statistik, Technische Universitaet Graz, September 2009.

[Tre96] Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, pages 103–120, 1996. 85

[YCW08] Dongkai Yang, Yi Chen, and Bisheng Wang. RFID Anti-Collision Algorithmin Logistics Service System. *IEEE*, 2008.

[YFh10] Chen Ying and Zhang Fu-hong. Study on Anti-Collision Q Algorithm for UHF RFID. In *International Conference on Communications and Mobile Computing*, pages 168–170. IEEE, 2010.

[ZCHG11] Yi Zhou, Kai Chen, Jianhua He, and Haibing Guan. Enhanced Slotted Aloha Protocols for Underwater Sensor Networks with Large Propagation Delay. *IEEE*, 2011.

[ZjTl09] Guo Zhen-jun and Huang Ting-lei. Study on Anti-collision Algorithms in UHF RFID System. In *Third International Conference on Genetic and Evolutionary Computing*, pages 458–461. IEEE, 2009.