



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

ASSIGNMENT OF MASTER'S THESIS

Title: Remote Keyless Entry Systems Security Analysis
Student: Bc. David Šafrata
Supervisor: Ing. Jiří Dostál, Ph.D.
Study Programme: Informatics
Study Branch: Computer Security
Department: Department of Information Security
Validity: Until the end of summer semester 2020/21

Instructions

Remote keyless entry systems are commonly used for accessing cars, buildings and other objects using radio frequency (RF) communication. Software-defined radios (SDR) are low cost and accessible devices to simply communicate on RF without any analog signal processing needed so they are used also in the field of cybersecurity to assess various RF protocols. RF transmission is prone to eavesdropping so there exist security protocols to ensure integrity and confidentiality. Analyze the most frequently used technologies and protocols used for remote keyless entry systems, e.g., rolling codes and KeeLoq. Do an in-depth analysis with a focus on implementation weaknesses, limitations and corner cases. Develop a demonstration platform using the KeeLoq development kit to support RF attack simulation. Test selected attacks using USRP software-defined radio.

References

Will be provided by the supervisor.

prof. Ing. Róbert Lórencz, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 13, 2020



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Master's thesis

Remote Keyless Entry Systems Security Analysis

Bc. David Šafrata

Department of Information Security
Supervisor: Ing. Jiří Dostál, Ph.D.

May 28, 2020

Acknowledgements

I would like to thank my supervisor for valuable insights and guidance during the work on this thesis. I would also like to thank my family for support during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No.121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 28, 2020

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2020 David Šafřata. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Šafřata, David. *Remote Keyless Entry Systems Security Analysis*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Tato diplomová práce se zabývá analýzou zabezpečení systémů bezklíčového vstupu. Obecné poznatky jsou rozšířeny o přehled několika systémů využívajících plovoucí kódy a jejich známých slabín. V dalších kapitolách je podrobněji analyzována bezpečnost systému KeeLoq, zejména se zaměřením na implementační slabiny. Tato detailní analýza vyústila v návrh nového útoku, který je schopen systém prolomit přibližně za jednu hodinu. Tento útok je spolu s dalšími vybranými útoky implementován za využití softwarově definovaného rádia USRP B210 a vývojového kitu systému KeeLoq. Závěr obsahuje autora bezpečnostní doporučení na základě nabytých znalostí.

Klíčová slova RKE, bezklíčové systémy, bezpečnost, bezdrátová komunikace, 433 MHz, plovoucí kód, KeeLoq

Abstract

This diploma thesis focuses on the security analysis of remote keyless entry systems. The general overview is followed by a summary of several systems that use the rolling code scheme and their known weaknesses. The KeeLoq system's security is analyzed more in-depth in the following chapters, primarily focusing on implementation weaknesses. This thorough analysis resulted in a description of a new attack that can exploit the system in approximately one hour. This attack, along with other selected attacks, is implemented using the software-defined radio USRP B210 and the KeeLoq system development kit. The conclusion contains the author's security recommendations based on gained knowledge.

Keywords RKE, keyless entry, security, wireless, 433 MHz, rolling code, KeeLoq

Contents

Introduction	1
1 Remote keyless entry systems	3
1.1 Authentication schemes	3
1.2 Operation modes	5
1.3 Attacks independent on specific system	6
2 Software-defined radios	11
2.1 Radio-frequency signals	11
2.2 Digital modulation methods	11
2.3 Data encoding	13
2.4 Operation of SDRs	14
2.5 USRP B210	14
2.6 Universal Radio Hacker	15
3 Approach	17
3.1 Direction and goal proposal	17
3.2 Platform choice	18
3.3 Implementation weaknesses	19
4 Specific rolling code RKE systems	21
4.1 AUT64	21
4.2 HITAG2	23
4.3 KeeLoq	24
5 KeeLoq cryptanalysis	29
5.1 Related work and considerations	29
5.2 KeeLoq cipher overview	30
5.3 Exhaustive search attack	30
5.4 Slide attack	31

6	Attacks on KeeLoq implementation	35
6.1	Altering the message	35
6.2	Bruteforcing the key derivation	36
6.3	Bruteforcing the encrypted part	37
6.4	Desynchronizing the counter	37
6.5	Breaking the counter	38
6.6	Side-channel analysis	38
7	New attack on KeeLoq counter	41
7.1	Introduction	41
7.2	Attack scope	41
7.3	Exploitable properties of the system	41
7.4	Attack description	42
7.5	Attack duration	44
7.6	Countermeasures	45
7.7	Future work	46
8	Demonstration platform	47
8.1	Common hardware and software setup	47
8.2	Features	48
8.3	Replay attack	48
8.4	RollJam attack	49
8.5	Counter breaking attack	49
9	Conclusion	51
	Bibliography	53
A	Acronyms	57
B	Contents of enclosed CD	59

List of Figures

1.1	Fixed code scheme	3
1.2	Rolling code scheme	4
1.3	Challenge-response scheme	5
1.4	Illustration of relay attack	7
1.5	Typical flow of RollJam attack	9
2.1	Some basic modulation schemes	12
2.2	Commonly used line codes	13
2.3	Schema of typical SDR operation	14
2.4	Part of the first screen of Universal Radio Hacker	15
2.5	The second screen of Universal Radio Hacker	16
3.1	KeeLoq GUI tool, for better insight into receiver's operation	18
3.2	RKE systems and the primary work focus path	19
4.1	One round od AUT64 block cipher	21
4.2	HITAG2 RKE message structure	23
4.3	KeeLoq RKE message structure	25
4.4	KeeLoq counter synchronization windows	26
5.1	Block diagram of KeeLoq encryption and decryption	30
5.2	Idea behind a typical slide attack	31
5.3	Illustration of meet-in-the-middle slide attack	33
7.1	Attack flow for implementations ignoring overflow bits	44
7.2	“Counter breaking” attack traversal	45
8.1	Common hardware setup of the demonstration platform	47
8.2	Terminal-based user interface	48
8.3	Button press simulator hardware	49

List of Tables

1.1	Overview of selected attacks on common RKE schemes	6
4.1	Use of selected RKE systems in automotive industry	22
4.2	KeeLoq key derivation schemes	25
5.1	Selected cryptanalytic attacks on KeeLoq cipher	29

Introduction

Remote keyless entry (RKE) systems became a common part of everyday life for many people. They are frequently used in car remotes, wireless garage door openers, and similar applications. As they are intended to protect access to valuable assets, while the radio frequency communication is prone to eavesdropping and other possible manipulations, they typically make use of cryptographic measures to provide security of the transmissions.

Remote controllers are usually low power battery-operated devices. Thus they commonly implement lightweight cryptosystems, which are usually less resource-heavy than cryptosystems used in traditional infrastructure. Some of them even use proprietary ciphers (KeeLoq, for example), which were subjected to cryptanalysis in several papers in recent years.

The first part of this thesis aims to make a complex analysis of used RKE systems and identify all possible weak spots, as they don't solely rely on the used cipher. General findings are applied in a more in-depth analysis of the KeeLoq system, focusing primarily on implementation weaknesses.

The second part of the thesis describes a new attack on the KeeLoq platform and presents a demonstration platform, which uses the KeeLoq development kit and the USRP B210 SDR to test selected attacks, including the new one.

Remote keyless entry systems

1.1 Authentication schemes

1.1.1 Fixed code

Fixed code was one of the first methods to provide remote keyless entry. Transmitter always sends the same data for a particular function of a system. This scheme doesn't provide any authentication (or security measures at all) and can be easily exploited by a simple replay of the signal. Some cars from notable automotive companies were equipped with RKE system based on fixed code scheme even around the year 2000 [1].

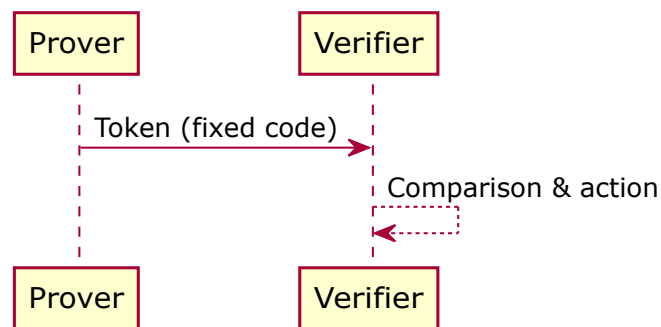


Figure 1.1: Fixed code scheme

1.1.2 Rolling code

Rolling code, sometimes called hopping code, is a common way of authentication in RKE systems. It is one-way authentication where a transmitter, sometimes called a prover, sends a message containing a nonce in its encrypted part to a receiver, sometimes called a verifier. The nonce assures that the mes-

sage is fresh, which provides replay attack protection. A value that serves as a nonce can be implemented in several ways.

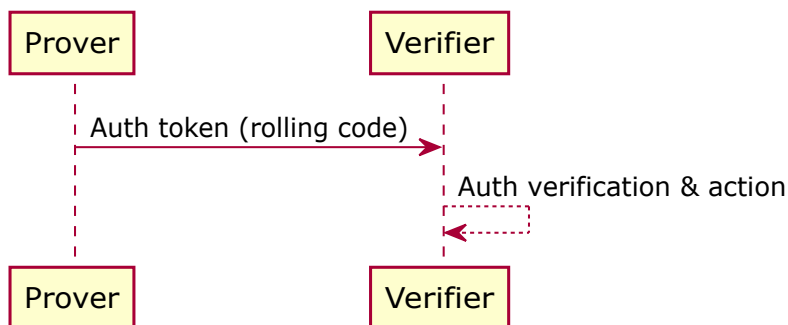


Figure 1.2: Rolling code scheme

1.1.2.1 Random or pseudo-random nonce

In general, it's a common way to generate a nonce with a cryptographically secure random number generator. However, to assure that the nonce is not reused, the verifier must hold a list of used nonces, which can lead to very high memory requirements, so it's not a typical approach in RKE applications.

1.1.2.2 Counter

A counter is one of the most straightforward ideas to easily provide non-repetitive value, where the next value is incremented by one in every use. The numbers below the current counter value are considered as used. Unlike the random nonce, it has constant memory complexity, as both sides (prover and verifier) only need to store the current counter value. This property brings the main downside of the counter method, which is the need to synchronize the value between both parties. The value stored in the verifier can be easily behind the value, as the radio frequency one-way transmission messages from prover may not reach the verifier, due to radiofrequency interference or being out of the range. Because of this property, typical implementations provide dropout tolerance. The counter method is commonly used in RKE applications, such as in HITAG2 and KeeLoq products [1, 2].

1.1.2.3 Timestamp

Another way to provide nonce is to use a timestamp. Value, which differs from the current time of a verifier, is rejected. The need for synchronized clock values requires a precise time source, typically achieved by using a real-time clock (RTC), which usually increases hardware complexity and costs. It

is commonly used in one-time password generators, but rarely in the field of RKE systems.

1.1.3 Challenge-response

If two-way communication is available, authentication can be provided by the challenge-response protocol. Upon request, a verifier sends a challenge to a prover, commonly containing a random sequence of bits. The verifier then uses a shared secret to prove its authenticity. The response to the challenge is provided by encrypting the challenge with a symmetric key or asymmetric private key or concatenating the challenge with a password followed by hashing. It may also combine mentioned approaches or use another similar one.

Either the authentication can be one-way, where the prover authenticates to the verifier, or it can be mutual, where the verifier also authenticates to the prover, in addition to the one-way authentication, to enhance the security of the system.

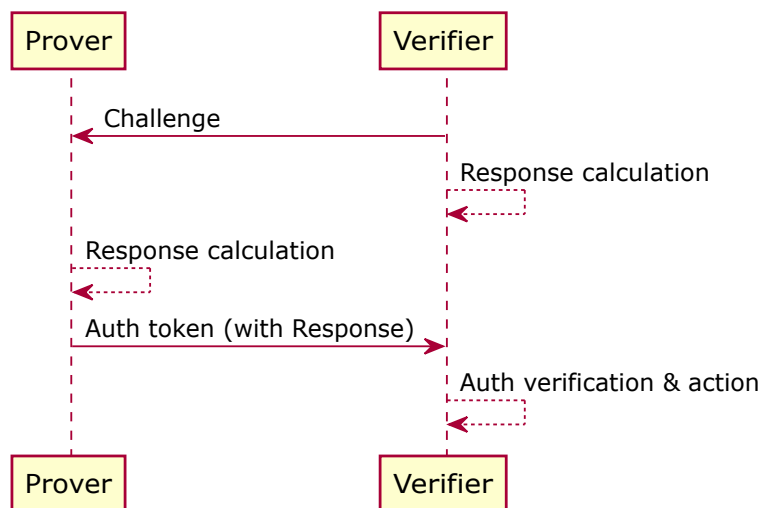


Figure 1.3: Challenge-response scheme

1.2 Operation modes

1.2.1 Active keyless entry

Active keyless entry systems are a widespread category of RKE systems. A keypress of a keyfob usually initiates the operation. It typically uses a rolling code scheme, but it may also use fixed code or serve as an initial message in a challenge-response scheme, in some cases.

The active property can restrict the number of attack scopes, as the only way to initiate communication is to press the button physically, so unless the attacker has the possession of a keyfob, the interaction with the transmitter is only limited to signal eavesdropping in case of one-way communication.

1.2.2 Passive keyless entry

Passive keyless entry systems are gaining more popularity in recent years [3]. It allows the user to use the RKE system without pressing any button, thus the name passive keyless entry. It is usually based on a challenge-response scheme.

When the user triggers the verifier (for example, by pulling the car door handle), the verifier sends a wake-up message with a challenge to the prover (for example in the form of a keyfob), which computes the response based on the shared secret and sends it to the verifier. So if the prover is close to the verifier, it provides authentication without any active manipulation with the prover device.

1.3 Attacks independent on specific system

Operation mode	Active	Active	Passive
Authentication	None/Fixed code	Rolling code	Challenge- resp.
Attack	Replay	RollJam	Relay
Impact	High	Moderate	High
User-detectability	Low to none	Low	Low to none
Complexity	Low	Medium	Medium

Table 1.1: Overview of selected attacks on common RKE schemes

1.3.1 Denial-of-service by RF jamming

Radio-frequency jamming is a simple attack, which disrupts the operation of a receiver very effectively. The receiver usually filters RF signals to accept only transmissions on a certain frequency. In order to receive the signal coming from the transmitter correctly, the transmission power of the signal should be on a certain level, where it's clearly distinguishable from the environmental noise. It is commonly referred to as a signal-to-noise ratio (SNR). The ratio should be greater than one for the clear reception of a signal.

The signal-to-noise ratio is usually tiny when the distance between transmitter and receiver is significantly long. It also decreases when noise levels spike. Such spikes are usually caused by other transmissions in the same RF

band, commonly called interference. These interferences may happen quite often, especially in ISM unrestricted bands, typically used by RKE systems. When interferences are intentional, it's usually called jamming. Devices intended to jam signal at a particular frequency just broadcast useless signals with high output power, which effectively increases the noise level, thus decreases the signal-to-noise ratio for other transmissions in reach. All wireless protocols, including the ones in RKE systems, are prone to this kind of attack based on simple physical foundations of the RF communication. Signal jamming is illegal in many countries [4].

The consequences of jamming attacks are questionable, as the RKE systems usually have mechanical alternatives to provide the same functions, like locking the car with a mechanical key or closing garage door with a physical button. In a sample scenario, when the attacker jams signal to close the car, it may only succeed if the RKE system user (i.e., driver) does not check the visual feedback (like a double blink of car lights) of the requested operation.

1.3.2 Replay/relay attack

A replay attack is probably the easiest when the attacker has hardware resources that allow capturing the signal, saving it, and transmitting it afterward. The growth of the SDR market and rising community involvement around cheap SDRs allowed almost anyone to dig into this area and perform similar attacks with minimal knowledge of underlying theory. A replay attack is sufficient to break fixed code RKE systems as the transmitted message for a particular requested operation is always the same.

The idea of rolling code systems was driven by the need to mitigate replay attacks [1], as the previously used codes were rejected by the receiver (more precisely, the verifier). Even though the rolling codes sufficiently tackled the problem of replay attacks, they remain prone to a more sophisticated replay attack variant called relay attack.

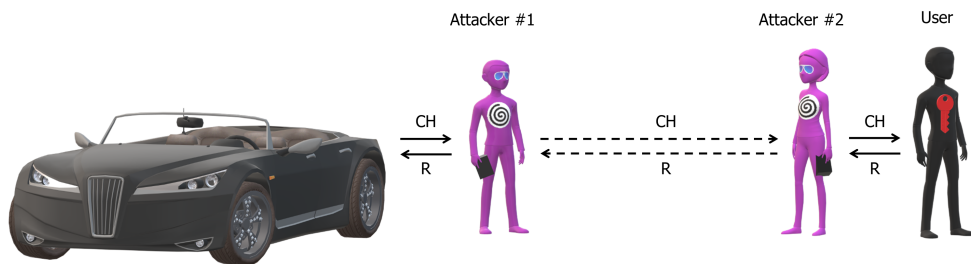


Figure 1.4: Illustration of relay attack on challenge-response passive keyless entry system

Relay attack is based on replay of signal from a transmitter located out of range of the receiver. When the attacker receives the signal from the transmit-

ter, it can be sent to the attacker's accomplice within the range of the receiver, which replays the message captured by the attacker. This setup works as an amplifier for the messages, which wouldn't go through to the receiver, or it can effectively work as a man-in-the-middle making this attack dangerous even for challenge-response based RKE systems. This statement supports research [3], which shows that relay attack is potentially perilous for a large amount of passive keyless entry systems, as the prover device requires no user interaction. Its operation is only based on the proximity of the keyfob, which could be easily bypassed by the relay attack, in the two attacker scenario (see Figure 1.4). Also, the paper shows that possible countermeasure options are limited.

1.3.3 RollJam attack

RollJam attack has been popularized by famous hacker Sammy Kamkar and presented in [5]. This attack combines jamming and replays to provide a way to break most rolling code RKE systems. The idea is based on the fact that the frequency bandwidth of the receiver is quite broad.

The jamming signal is transmitted at frequency $f + \Delta$, where the f is the center frequency of the attacked RKE system. The reception of the signal is set to frequency f , but is set to accept only frequencies in a narrow band around frequency f , to filter out the jamming signal at frequency $f + \Delta$, which is intended only to jam the legitimate receiver.

At the beginning of the attack, the attacker starts with signal jamming. When the attacker accepts the first message, it's saved until it receives the second message, which is also saved. Right by that time, the attacker stops jamming and replays the first message which didn't go through at the time of its original transmission because of the jamming signal interfering with the receiver. The legitimate user thinks that the second message went through as intended, but it's actually the first one accepted by the receiver. It allows the attacker to replay the second message later as its counter is still fresh for the receiver counter value, as the last message delivered contained counter lower by one.

The practical impact of the attack is quite limited. Firstly, the second saved message becomes useless as other messages with the following counter values are transmitted as the value of the saved message becomes non-fresh. Secondly, even if the attacker obtains two following messages, it is rarely applicable to exploit the subject protected by the RKE system. For example, when the attacker executes the attack in a parking lot when someone closes a car via the RKE system, it would only allow the attacker to close the car again, posing no threat. When someone opens the car, it may allow the attacker to open it once again, but it would be probably useless as the legitimate user usually drives away right after and then close it again via the RKE later, making the saved message useless.

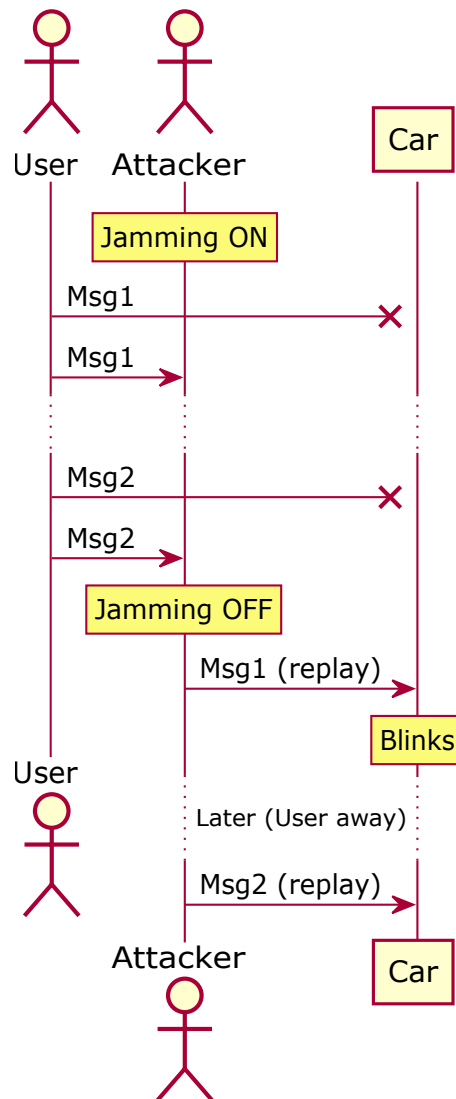


Figure 1.5: Typical flow of RollJam attack

One possible practical scenario could be with garage door openers when the driver uses the RKE system to open the garage door and then closes it with a physical button in the garage once the driver exits the car. In this case, the attacker would be able to open the garage door later. Another possible scenario would be, when a car user opens a trunk by the RKE, as it typically requires no further RKE messages as the car closes the central lock when the trunk door is mechanically closed. So again, the attacker can access the car later through the trunk door.

Software-defined radios

2.1 Radio-frequency signals

A signal can be described by the following function over time (t)

$$S(t) = A \times \sin(2\pi \times f \times t + \phi) \quad (2.1)$$

Where A is the amplitude of the sine wave, f is the frequency, and ϕ is the phase shift. The basic modulation methods are based on modifying one of these components over time.

The modern signal processing works with quadrature signals, sometimes called I/Q signals. They consist of two paired signals – I (in-phase) and Q (quadrature) containing information of a reference signal. The Q signal is shifted by 90 degrees relative to I signal.

Quadrature modulator is a circuit used for generating RF signals based on I/Q signals. It consists of 2 frequency mixers, where one mixes the local oscillator (LO) with the I signal, and the other one mixes the LO shifted by 90 degrees with the Q signal. The output of these two mixers is then summed, resulting in the desired RF signal. Quadrature demodulator works on the same principle, but the signal flow is backward.

2.2 Digital modulation methods

2.2.1 ASK

In amplitude-shift keying (ASK), the signal is modulated by changing the carrier wave's amplitude over time.

$$S_{ASK}(t) = A(t) \times \sin(2\pi \times f \times t + \phi) \quad (2.2)$$

For binary ASK, 1 is represented by amplitude A_1 , and 0 is represented by amplitude A_2 . There is a particular variation of binary ASK called on-off

keying (OOK), where A_2 is zero. It is used in many applications as it's easy to implement.

2.2.2 FSK

In frequency-shift keying (FSK), the signal is modulated by changing its frequency over time.

$$S_{FSK}(t) = A \times \sin(2\pi \times f(t) \times t + \phi) \quad (2.3)$$

For binary FSK, 0 is typically represented by $f - \delta$, and 1 is typically represented by $f + \delta$, where f is the center frequency of the carrier wave.

2.2.3 PSK

In phase-shift keying (PSK), the signal is modulated by changing the phase of the carrier wave over time.

$$S_{PSK}(t) = A \times \sin(2\pi \times f \times t + \phi(t)) \quad (2.4)$$

For binary PSK, 0 is represented by phase shift ϕ_1 , and 1 is represented by phase shift ϕ_2 , typically 180 degrees apart.

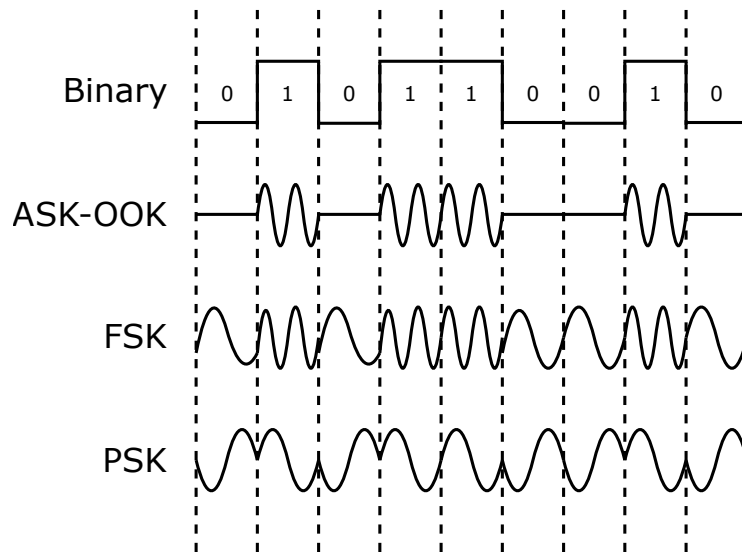


Figure 2.1: Some basic modulation schemes [6]

2.2.4 Complex modulations

For higher data rates, more complex modulation methods may be used. It can be improved by encoding more bits to one state of the particular scheme.

For example, quadrature PSK (QPSK) can be represented by 4 constellation points, which means 4 possible combinations of 2 bits, by 4 distinct phase shifts (0, 90, 180 and 270 degrees). Also, it is possible to combine several modulation methods. For example, the newest WiFi standard (802.11ax) uses 1024-QAM modulation [7], which combines ASK and PSK, providing 1024 constellation points, which allows encoding 10 bits of data at a time.

However, these complex modulations are rarely used in RKE applications. The remotes are usually low-cost battery-operated devices, so they usually use one of the binary modulation methods, for example, ASK-OOK.

2.3 Data encoding

Transmitted binary data needs to be encoded before they are transmitted over the air. There are several encoding schemes, which are commonly used. They are called line codes, as they were designed for transferring digital data on telecommunication transmission lines. Most common codes include Non-return-to-zero (NRZ), Return-to-zero (RZ), Manchester (Biphase-L), Differential Manchester (Biphase-M, Biphase-S and D. M.) and Bipolar code. [8] Sometimes other codes may be used, like PWM in case of KeeLoq [9].

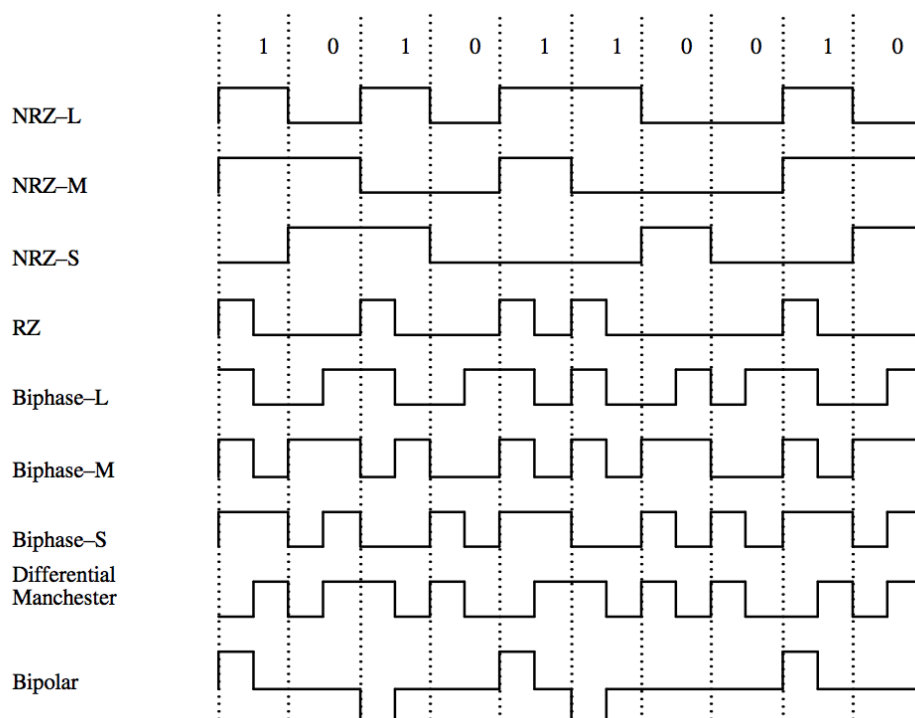


Figure 2.2: Commonly used line codes [8]

2.4 Operation of SDRs

The fundamentals of operation of common SDRs are quite straightforward, as all the signal processing is provided in the software. In simple terms, SDRs provide analog/digital (A/D) and digital/analog (D/A) conversion capabilities applied for RF signals.

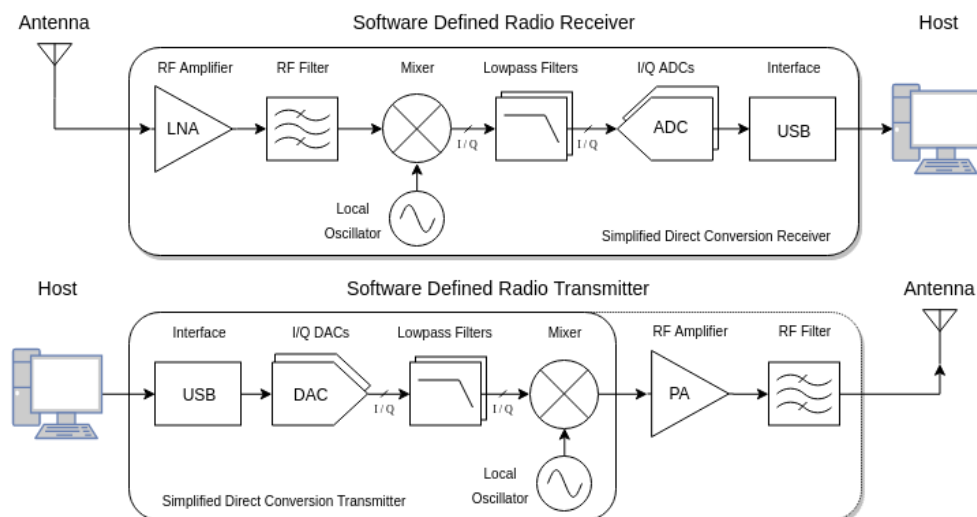


Figure 2.3: Schema of typical SDR operation [10]

For the reception, a signal received on the antenna is filtered, mixed with the local oscillator (LO), set to the desired center frequency, and produced I and Q signals are filtered and sampled by A/D converters delivering a stream of digital sample values. Sampled data are then passed to PC for further processing, for example, via the USB interface. For transmission, the flow is similar but goes backward. The digital samples of I and Q signals go to D/A converters, and then they are filtered and mixed with LO producing desired output signal, which is then amplified, filtered, and transmitted by the antenna. [10]

2.5 USRP B210

USRP B210 is a high-end SDR made by Ettus Research, which operates in the frequency range from 70 MHz to 6 GHz. It allows connecting up to four antennas, two for a reception, and the other two for both reception and transmission. The SDR supports bandwidth up to 56 MHz. The hardware is based on Xilinx Spartan 6 XC6SLX150 FPGA. [11] USRP B210 is supported by open-source SDK called USRP Hardware Driver (UHD), produced by the manufacturer. It also provides bindings to other popular APIs, like GNU Radio.

2. SOFTWARE-DEFINED RADIOS

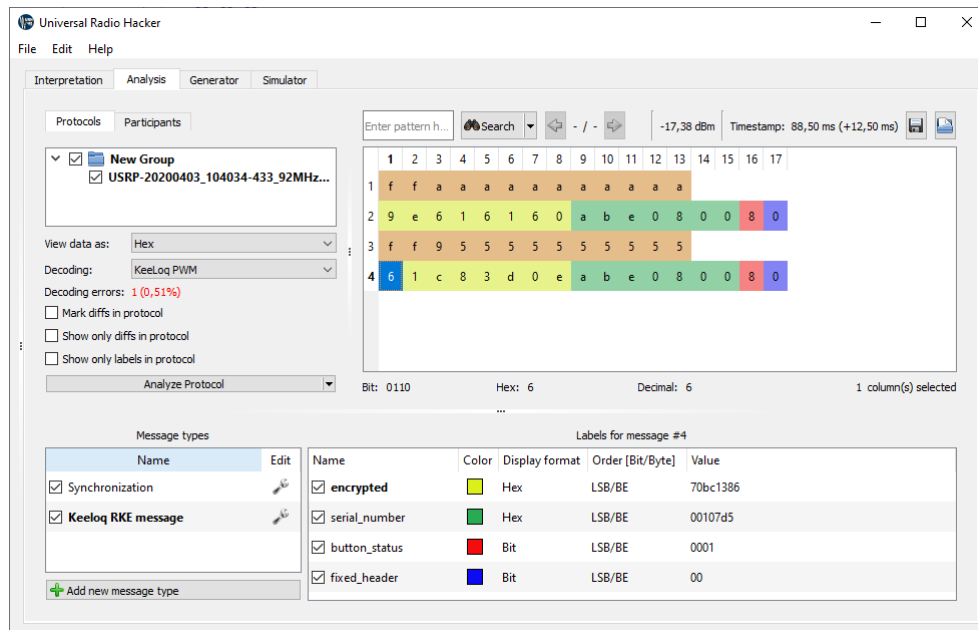


Figure 2.5: The second screen of Universal Radio Hacker, showing KeeLoq capture with two labeled messages

The fourth screen is intended to make simulations for more complex communication schemes. It allows the user to prepare message logic flow, which can be used for attack, where the SDR should send messages according to received messages. This screen was used neither for the analysis, nor attack demonstrations.

Approach

3.1 Direction and goal proposal

As the title of this thesis suggests, the goal of this thesis is to provide a security analysis of RKE systems. This task could be handled in several ways. It can be either shallow analysis covering a vast amount of systems, making an overview of state of the art. It can also be work that would cover only some technologies with a more in-depth focus on one particular system. Nevertheless, doing a thorough analysis of multiple RKE systems is beyond the scope of this thesis.

A thesis focused on summarizing the work done in the whole field has its significance, but mostly it won't produce any novel approaches or findings. On the other hand, the thesis, which concentrates on one topic may provide some new exciting outcomes due to an increased time frame for the in-depth analysis but also may fail to do it. This approach's results could be precarious as sometimes when similar work is done to find any new vulnerability of a system doesn't produce any new findings. In any case, when the work describes the effort methodically, it could be beneficial for any following effort, as other researchers can avoid "dead alleys". But, when some newly discovered approach or idea is proven to have some significance, it could have a tremendous further impact.

In order to produce some possibly beneficial results, only one path was chosen to follow with more in-depth analysis and implementation work. From the overview of currently known generic attacks (see Table 1.1) it can be observed that only authentication or operation scheme without potent generic attack is a rolling code scheme. Thus, the rolling code appears to be the most interesting scheme from the viewpoint of the system's overall security.

Therefore, the following chapters for mentioned reasons only focus on the systems using the rolling code scheme, where some specific systems are introduced in Chapter 4. The choice of the particular platform for more profound analysis is explained in the following section.

3. APPROACH

The goal is to analyze common rolling code RKE systems and their known security flaws. Then, focusing on one specific system, research relevant cryptographic attacks, and analyze implementation vulnerabilities and design flaws. In the case of discovering a new weakness, it should be described in detail to provide valuable results. Using the SDR and development kit of the selected platform, some attacks should be tested in a replicable way in the form of an HW/SW demonstration platform.

3.2 Platform choice

After extensive analysis of possible systems to choose from available on the market, it was decided to go further with the KeeLoq rolling code RKE system. There were several reasons behind this selection, which are described in the following paragraphs.

The products based on the KeeLoq system as the technology itself are provided by Microchip, which is a well-established player in the semiconductor market. There are several development kits, which can be purchased easily through standard sale channels. The chosen development kit, sold as DM182017-4 is provided with plenty of documentation and also has decent PC software support [2].

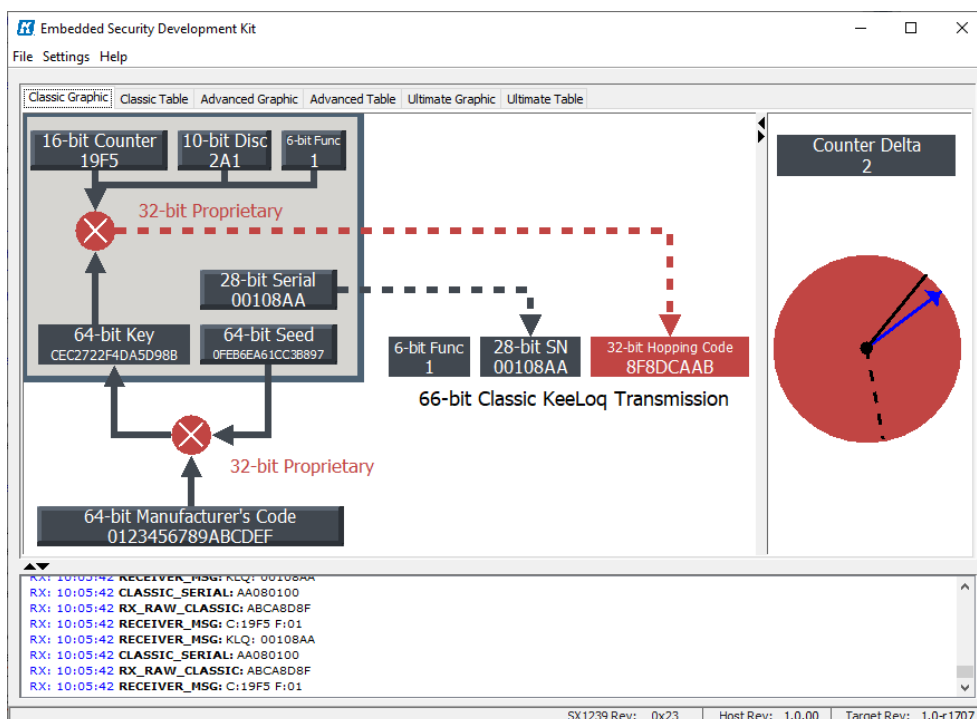


Figure 3.1: KeeLoq GUI tool, for better insight into receiver's operation

The provided PC tool allows the researcher to have better insight into the system’s internal operation, especially on the receiver side. As this system is well-documented in several datasheets, it allows visualizing the operation covered in these sources, making the work on this system more user-friendly.

As Table 4.1 shows, the KeeLoq system is used by several automotive manufacturers for RKE systems and may be used in garage door openers and similar applications [9].

Also, the amount of research work around KeeLoq suggests that this platform has its significance. However, most of the work done on this topic is focusing predominantly on the cipher itself. It leaves space for a more sophisticated approach or more precisely to focus on implementation weaknesses, which is often neglected in these papers.

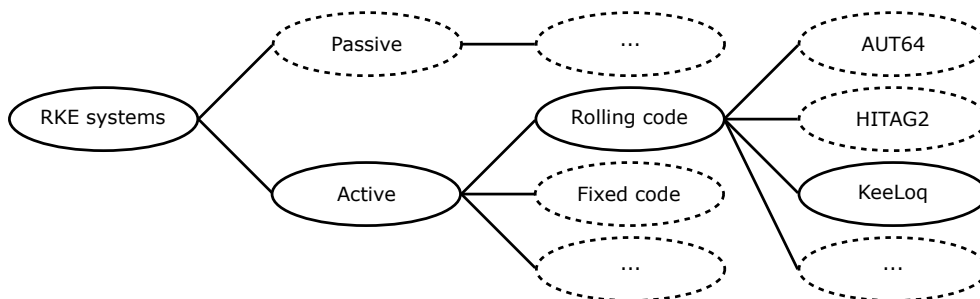


Figure 3.2: RKE systems and the primary work focus path
(Full ellipses – primary focus, dashed ellipses – secondary focus)

3.3 Implementation weaknesses

“A chain is no stronger than its weakest link.” — proverb

When the perspective on the system is complex, and the analysis tries to assess the weakest spot, the cryptanalysis of used cipher and particular implementation flaws have the same significance on the system’s overall security, as either one of them can be the weak point.

The whole security field is based on preventing unauthorized use or access, so research of flaws for a particular system should be made from a potential attacker’s point of view. It means it’s necessary to pay the same attention to vulnerabilities of a system and tied ecosystem as to underlying mechanisms, such as used cipher.

Specific rolling code RKE systems

4.1 AUT64

AUT64 is a proprietary cipher, which is predominantly used in the automotive area. One research [1] shows that this cipher is, for example, used in RKE rolling code systems called VW-2 and VW-3, used by manufacturers in automotive group Volkswagen, which has significant market share worldwide.

This iterative cipher with an unbalanced Feistel scheme operates over 64-bit blocks and employs key-dependent permutation and round function. Volkswagen RKE system implementation uses 12 rounds, despite recommended 8 or 24 rounds, and the key is 120-bit. The design limitations of the permutation function and the S-boxes (used in the round function), limit the key entropy to approximately 88 bits of information. [12]

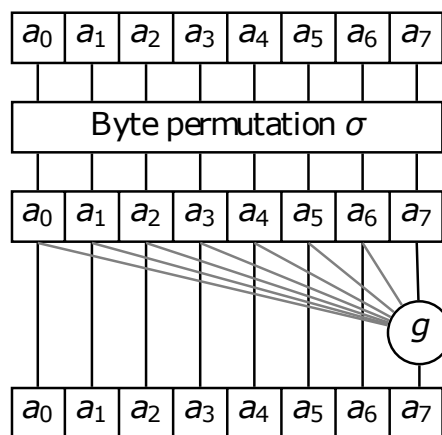


Figure 4.1: One round of AUT64 block cipher [1]
(g is the round function, a_{0-7} is 8-byte (64-bit) block)

4. SPECIFIC ROLLING CODE RKE SYSTEMS

The best-known attack [12] on an 8-round AUT64 variant, which needs 16 chosen-plaintext pairs, has complexity equivalent to $2^{50.7}$ encryption operations, which can be, according to authors, broken in milliseconds. The 24-round variant used in Atmel’s TK5561 system hasn’t been broken by conventional cryptanalysis yet. However, the embedded weak key management of this particular implementation makes the system practically broken [12].

For the non-standard 12-round AUT64 variant used by the Volkswagen group, [1] shows, that the manufacturer uses no key diversification, which means, that all cars with VW-2 and VW-3 systems each use global encryption key, which means that every car using particular system uses the same key. This weak key management crushes the overall security of the used system, regarding the means like reverse engineering of ECU firmware or side-channel analysis of transponder encryption operation, which can effectively recover the key.

Car brand	AUT64	HITAG2	KeeLoq	Reference
Alfa Romeo		•		[1]
Audi	•			[1]
Citroen		•		[1]
Dacia		•		[1]
Daewoo			•	[13]
Fiat		•	•	[1], [13]
Ford	•	•		[12], [1]
GM			•	[13]
Honda			•	[13]
Chrysler			•	[13]
Jaguar			•	[13]
Lancia		•		[1]
Mazda	•			[12]
Mitsubishi		•		[1]
Nissan		•		[1]
Opel		•		[1]
Peugeot		•		[1]
Proton	•			[12]
Renault		•		[1]
Seat	•			[1]
Škoda	•			[1]
Toyota			•	[13]
Volkswagen	•		•	[1], [13]
Volvo			•	[13]

Table 4.1: Use of selected RKE systems in automotive industry

4.2 HITAG2

HITAG2 is another cipher commonly used in rolling code RKE systems. It is used across various automotive vendors, typically using the NXP microcontroller series PCF7946 and PCF7947 based on its reference implementation. The car series using HITAG2 for their RKE systems include Opel, Renault, Peugeot, Fiat, Nissan, and many other vendors. [1]

This cipher is a stream cipher based on 48-bit LSFR and non-linear function with 20 bits on input and one bit on output, XORed with part of key and initialization vector stored in their respective shift registers, feeding the LSFR. The non-linear function's output produces 32-bits of the keystream, which is used for authentication of the token. The initial state of the LSFR consists of 32 bits of the serial number, lower 16 bits of the encryption key. The remaining 32 bits of key, meaning the HITAG2 uses 48-bit keys, are in the 32-bit shift register. The other 32-bit shift register with IV is initiated with the 28-bit rolling code counter and 4-bit button status. [14]

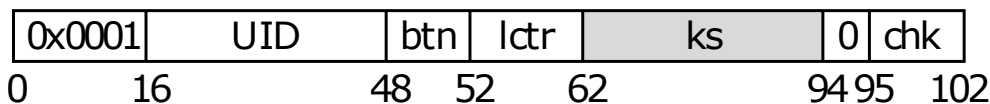


Figure 4.2: HITAG2 RKE message structure [1]

The message (see Figure 4.2) starts with 16 bits of fixed-value synchronization header, and then there are 32 bits of the serial number (UID), 4 bits of button status (btn), 10 lowest bits of the 28-bit counter (lctr), followed by 32 bits of keystream (ks) for authentication and 8 bits of the message integrity checksum (chk).

The best-published attack [1] before 2018 used key-guessing based on correlation score, calculated for a small number of keystreams extracted from eavesdropped messages. This attack also needs to know the initial value of the IV register, which contains 28 bits of the counter, while the plaintext part of the authentication token contains only 10 lower bits of the counter. However, as the counter starts from 0, the upper bits are usually predictable (based on vehicle age) from the interval of 10-bit overflows of the exposed counter value part, which is usually from 0 to 10. With correctly guessed upper bits of the counter, researchers claim to be able to break this cipher within one minute on an ordinary laptop with a relatively small amount (4 to 8) of captured messages.

In 2018, there was published new research [15], which presented so far, the quickest way to retrieve the key, based only on the retrieved 32-bit keystream. The attack is based on guess-and-determine principle, which exploits leakage of information from the state register to the keystream. With the presented optimized GPU implementation of the attack, it is possible to fully recover

the key on consumer-grade PC in just 1.2 minutes, which makes this system completely broken.

4.3 KeeLoq

4.3.1 System overview

KeeLoq is an RKE system manufactured by company Microchip. For clarification, this section describes the original KeeLoq system, which is now marketed under the name *KeeLoq Classic* and is the one referred across this thesis. The Microchip recently introduced the next-generation KeeLoq system under the name *KeeLoq Ultimate*, which provides multiple ways to strengthen the overall security of the system and uses widely-used AES-128 cipher, instead of proprietary KeeLoq cipher. [9]

The structure and properties of the proprietary KeeLoq cipher used in the KeeLoq system are described in 5.2.

4.3.2 RF properties

The KeeLoq can operate on a wide range of RF bands from 315 MHz to 915 MHz, including ISM band 433.92 MHz, which is used in the reference implementation in the development kit. It supports two kinds of modulation – FSK and ASK-OOK, also used in the reference implementation. [9]

KeeLoq supports either Manchester encoding (Biphase-L), or PWM encoding (where $110 \approx 0$ and $100 \approx 1$), which is used in the reference implementation.

The experiments made on the KeeLoq development kit show that the receiver can process signals with center frequency varying from 430.42 MHz to 437.42 MHz, which is ± 3.5 MHz from the center transmission frequency of 433.92 MHz. This observation is interesting for attacks utilizing jamming like RollJam (see 1.3.3).

4.3.3 Message structure

As the system is based on a rolling code scheme as other systems described in this chapter, the communication consists only of one type of message under regular operation. The transmitter also supports sending special seed message, which is sent only during secure learn (pairing) procedure, typically only at the beginning of the transmitter's lifetime.

The message is always preceded by a short synchronization sequence of alternating 1s and 0s. The message itself (see Figure 4.3) starts with two 0 bits, which is followed by 4 bits of button status, where each bit corresponds to a particular button pressed. Then the message contains the 28-bit serial number, followed by 32 bits of the encrypted part of the message. The encrypted

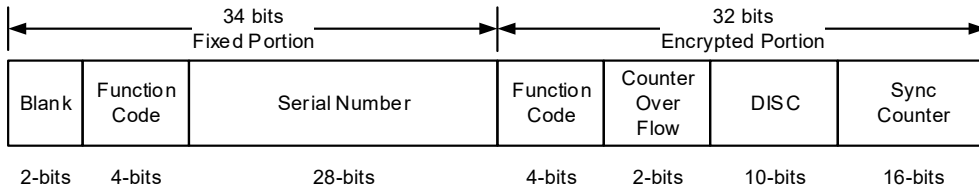


Figure 4.3: KeeLoq RKE message structure [9]

part contains 4 bits of button status, 2 bits signaling buffer overflows of transmitter counter, 10 bits of fixed discrimination value (typically lowest 10 bits of the serial number), and 16-bit counter value.

4.3.4 Key derivation

The encrypted part is encrypted by KeeLoq cipher using a 64-bit device key. All key derivations employ the master key, typically shared among the same manufacturing series, as the master key used for the derivation of the device key, must be stored in the receiver's memory. This feature allows the receiver to pair any transmitter from the same series by deriving the device key using the master key. The device key is generated with the same master key during the manufacturing process and stored in the transmitter's memory.

Random bits	Upper 32-bit seed part	Lower 32-bit seed part
0	$0100_2 SN_{27...0}$	$0010_2 SN_{27...0}$
32	$0000_2 SN_{27...0}$	$RS_{31...0}$
48	$0000_2 SN_{27...16} RS_{47...32}$	$RS_{31...0}$
60	$0000_2 RS_{59...32}$	$RS_{31...0}$

Table 4.2: KeeLoq key derivation schemes [16]
(SN - serial number, RS - random seed)

KeeLoq supports 4 possible key derivation schemes [17]. The first one, called *Normal learn* is based solely on the serial number. This type of derivation is user-friendly as, during pairing, there is no need to send seed packets, which is usually initiated by pressing all buttons at one time. However, when the master key is exposed, i.e., by side-channel analysis (see Section 6.6), the whole system is completely broken as the attacker needs only one captured message to derive the device key of the corresponding transmitter. Then the attacker can produce any valid message indistinguishable from the exploited transmitter's message, transmitted to the receiver by SDR or similar device, breaking the system's security completely.

Other 3 derivation schemes are covered under *Secure learn*. It means that this kind of the key derivation provides additional security to *Normal learn*, as the key derivation is seed-derived, where the seed is present only

in special packets, thus unknown to eavesdropper of regular communication. KeeLoq documentation specifies 3 distinct seed sizes: 32 bit, 48 bit, and 60 bit. The seed packet has a similar structure as a regular packet, only the seed bits replace the respective part (from the LSB) of the serial number, and encrypted part combined to a 60-bit chunk. This chunk is then padded with 4 zeros on the MSB side and split into two parts of the seed. These two parts are either decrypted by the master key or XORed with it. The two 32-bit results of the operation are then concatenated, which forms the 64-bit device key. However, as the attack described in Section 6.2 shows, the 32-bit and 48-bit seed variants cannot be considered *secure* at all, as it can be broken by brute-force attack in a relatively short time.

4.3.5 Authentication and replay attack countermeasure

The encrypted part provides authentication of the message itself as well as replay attack protection, by using the 16-bit counter as a nonce. The transmitter counter is incremented with every transmission, and its new value is saved to the memory of the receiver in order to check the validity of counter values contained in the following received messages. The authentication is provided by comparing the button status in encrypted part to the plaintext part button status, and by checking the discrimination bits, which is usually 10 least significant bits of the serial number, which is also present in the plaintext part of the message.

The one-way communication property of the rolling code implies that some messages transmitted may not reach the receiver, without any detection. Thus, the receiver side has a window of a few following counter values to be accepted, in case some messages were lost in the air.

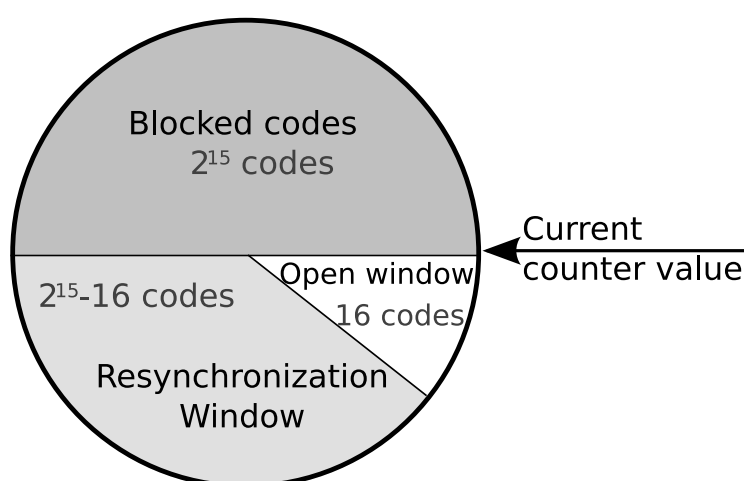


Figure 4.4: KeeLoq counter synchronization windows [16]

KeeLoq divides the counter period to 3 windows (see Figure 4.4) relative to counter value saved in the receiver. The first window has a size of 16 following values, meaning if the next received counter value contained in the message falls into this window, the operation is performed right away, and the receiver's counter is set to this new value. The second window follows the first window and has a size of $2^{15} - 16$. The receiver acts and moves its counter only if two directly consecutive messages fall into this second window. The last third window covers the remaining 2^{15} counter values. When the counter contained in the received message falls into this window, it's immediately rejected as it's considered as a replay of some previously transmitted message.

KeeLoq cryptanalysis

5.1 Related work and considerations

In recent years, there have been published numerous articles focusing on cryptanalysis of the proprietary KeeLoq cipher used in products of the same name. In the following sections of this chapter, there are selected only the most promising ones, which conform criteria of the required codebook size, limited by possibly available plaintext-ciphertext pairs. In the RKE mode of the KeeLoq system, at least 10 bits of the plaintext are always fixed (discrimination bits). So the amount of possible pairs is bounded from above to 2^{22} .

Yet, still, most of the published attacks have minimal application for KeeLoq use in rolling code RKE setup, as the attacker has access typically only to ciphertext, which can be, unlike plaintext, eavesdropped from the wireless communication. A lot of published attacks require 2^{32} known-plaintext pairs, which is the complete codebook, so unless the attacker needs the exact key for further processing, it has no practical value, as the attacker has access to any plaintext corresponding to ciphertext and vice versa. As a matter of interest, the best attack published to this day has a complexity of $2^{28.7}$ KeeLoq encryptions and slightly over 16 GB of memory based on a slide attack and fixed-points and requires the whole codebook.

Attack	KP pairs	Complexity	Reference
Exhaustive search	1 – 2	2^{64}	–
Slide/Algebraic	2^{16}	2^{53}	[18]
Slide/MITM	2^{16}	$2^{44.5}$	[19]
Slide/Fixed points	2^{32}	$2^{28.7}$	[18]

Table 5.1: Selected cryptanalytic attacks on KeeLoq cipher
(*Complexity* – is time complexity in equivalent of KeeLoq encryptions)

5.2 KeeLoq cipher overview [20]

Although the KeeLoq cipher diagram (see Figure 5.1) may resemble a typical NLFSR-based stream cipher, it's actually block cipher with highly unbalanced Feistel scheme. The unbalanced property comes from only one bit at one side in each round. KeeLoq cipher has 32-bit block size and 64-bit key length. Both encryption and decryption consist of 528 rounds, which is not divisible by 64, which is the length of the period of the key register. This property should make slide attacks (see 5.4) more difficult.

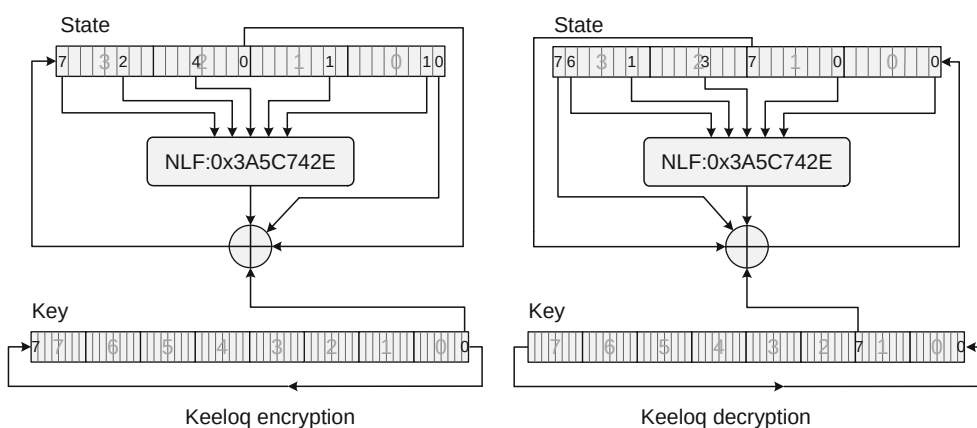


Figure 5.1: Block diagram of KeeLoq encryption and decryption [21]

For encryption, in each round, the 32-bit state NLFSR register is fed back to the most significant bit by XOR of the output of bits 0 and 16 of the state register, bit 0 of the key register, and the non-linear function 0x3A5C742E with bits 1, 9, 20, 26 and 31. The key register is rotated right by one bit in every round.

For decryption, in each round, the state register is fed back to the least significant bit by XOR of the output of bits 15 and 31 of the state register, bit 15 of the key register, and the non-linear function 0x3A5C742E with bits 0, 8, 19, 25 and 30. The key register is rotated left by one bit in every round.

5.3 Exhaustive search attack

Exhaustive search or brute-force attack is the main obvious point to start. In the case of KeeLoq cipher, as the key has 64 bits, the worst-case scenario has a complexity of 2^{64} KeeLoq encryptions, as the attacker needs to verify all possible keys. On average, the attacker would find the key after trying half of the possible key values, meaning the average complexity is 2^{63} . The advantage of exhaustive search is that it's "embarrassingly parallel problem" by design, which means that trying particular keys is not dependent on other tries, and the operation can be parallelized to numerous cracking instances. So the time

requirements of the exhaustive could be significantly reduced by using FPGA or GPU, comparing to sequential CPU calculations.

The complexity of the brute-force attack is the essential benchmark for all other attacks. Sometimes it's possible to find some new cryptographic attack, but its complexity is higher than the exhaustive search. Such an attack is unusable, but it can affect other research work, by showing which cryptanalytic approaches don't suit well the cipher.

5.4 Slide attack

The main idea of slide attack is to reduce the complexity of ciphers with the self-similar structure of their operation. It is commonly used for cryptanalysis of ciphers with a large number of rounds, usually done to hinder traditional cryptanalytic approaches, like differential cryptanalysis. This attack needs to operate as a known-plaintext attack, and the cipher needs to be able to break down to several iteration steps of identical function F .

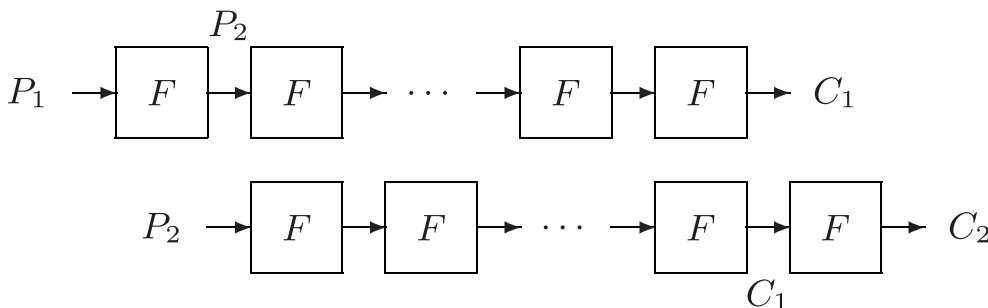


Figure 5.2: Idea behind a typical slide attack [19]

In order to successfully execute the slide attack, a slid pair must be found. It's a pair of two plaintext-ciphertext pairs (P_0, C_0) and (P_1, C_1) , which satisfies condition, that $(F(P_0), F(C_0)) = (P_1, C_1)$, which means, that this pair is shifted or slid by one iteration of the function F . Once this pair (P_S, C_S) is found the attack on the whole cipher can be simplified to break single iteration of the function F , as the slid pair serves as a pair of known plaintext-ciphertext $(P_0, P_1) = (P_S, C_S)$ for the function F .

Finding a slid pair is not an easy process, but with the birthday paradox, (similarly applied to search for hash function collisions), the amount of required plaintext-ciphertext pairs is limited to the square root of the codebook size. In the case of KeeLoq cipher, it means, that statistically, a slid pair should be found in 2^{16} plaintext-ciphertext pairs.

KeeLoq cipher does not conform to the general attack prerequisites, as the number of rounds (528) is not divisible by 64, which is the size of the key and its shift register period. Following attacks cope with the remaining 16 rounds specifically to reduce the number of rounds to attack to 512, which

can effectively transform the problem to be solvable as series of 8 iterations of 64 cipher rounds to satisfy the slide attack requirements.

5.4.1 Slide/Algebraic attack

The best algebraic attack [18] using the sliding property, can successfully recover the key in an equivalent of 2^{53} KeeLoq encryptions. In order to successfully execute the attack, one slid-pair over 64 rounds of KeeLoq encryption of function F needs to be found. Once the slid-pair (P_0, P_1) is found their corresponding ciphertexts (C_0, C_1) , can be also considered as slid pair of function F , but with key rotated by 16 positions, due to extra 16 rounds denoted as function G (see Equation 5.1).

$$E_k(x) = G_k(F_k^{(8)}(x)) \quad (5.1)$$

Authors of the paper used MiniSat SAT solver for 64 rounds of KeeLoq described by boolean equations, both for regular 64 rounds of the encryption and 64 rounds with shifted key by 16 positions, to recover the key algebraically. Without guessing any key bits, it can be done in 2^{32} CPU clocks. This procedure is applied for a significant number of possible plaintext pairs or slid-pair candidates in order to find the slid-pair, which should be found in 2^{32} slid-pair candidates generated from 2^{16} known-plaintext pairs.

Overall complexity is $2^{32+32} = 2^{64}$ CPU clocks, which is equivalent to 2^{53} KeeLoq encryptions.

5.4.2 Slide/Meet-in-the-Middle

Understanding of the attack description depends on familiarity with *linear step*, first introduced by Bogdanov's paper [20]. It's a way how to recover key bits when there are two states separated by 32 rounds at maximum. Due to the cipher structure, the transition between these two states is dependent only on particular key bits, which can be recovered in linear time.

The attack works with the 64-bit key divided to 4 equally-sized parts by 16 bits $k = (\hat{k}_3, \hat{k}_2, \hat{k}_1, \hat{k}_0)$. In the beginning, lowest 16 key bits ($= \hat{k}_0$) are guessed. This allows to partially encrypt plaintexts and partially decrypt ciphertexts, from the known-ciphertext pairs, using 16 rounds of the KeeLoq cipher with the guessed k_0 part of the key.

Then lower 16 bits of P_j^* , which denotes result of partial decryption of plaintext P_j by \hat{k}_3 , are guessed. For each guess, the \hat{k}_3 part of the key is determined by the linear step. Using the determined \hat{k}_3 , Y_j is partially decrypted to value Y_j^* and the upper 16 bits of P_j^* are known from the P_j . The tuple of $(P_j^*, Y_j^*, \hat{k}_3)$ is then saved to the hash table for future recovery.

As the upper 16 bits of X_i^* are equal to lower 16 of P_j^* (by the structure of the cipher) and X_i generated in the beginning, it is possible to recover \hat{k}_1 part

of the key by linear step for each plaintext P_i . With \hat{k}_3 it is possible to partially encrypt C_i to C_i^* . Then as the upper 16 bits of the C_i^* should be equal to lower 16 bits of Y_j^* . It is then possible to look for such collisions to values contained in the hash table. Once the collision is found, \hat{k}_2 is determined by linear step from X_i^* and P_j^* and \hat{k}_2' same way from C_i^* and Y_j^* . If the \hat{k}_2 is equal to \hat{k}_2' , it has high likelihood (it can be checked), that the P_i and P_j is a slid pair, as they satisfy the slide property for the function F of 64 KeeLoq rounds with recovered key $k = (\hat{k}_3, \hat{k}_2, \hat{k}_1, \hat{k}_0)$.

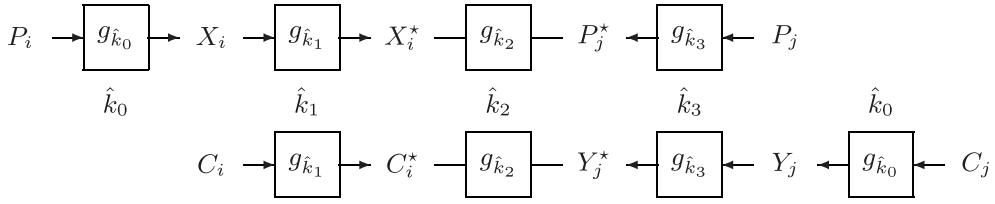


Figure 5.3: Illustration of meet-in-the-middle slide attack [19]

According to the authors' calculations, the complexity of this attack is equivalent to 2^{45} KeeLoq encryptions. A generalized form of this attack has even slightly reduced complexity, equivalent to $2^{44.5}$ KeeLoq encryptions. The memory complexity of both variants is negligible, in lower units of MB.

Attacks on KeeLoq implementation

6.1 Altering the message

This diploma thesis focuses on the security analysis of remote keyless entry systems. The general overview is followed by a summary of several systems that use the rolling code scheme and their known weaknesses. For experiments described in the following two subsections, there was used Python 3 script, which produced arbitrary messages based on input data, like counter value, function bits, overflow bits, and serial number. The generated data were written into XML, which were after that transmitted via Universal Radio Hacker, with USRP 210 SDR connected.

It allowed instantiating a virtual transmitter with a serial number of a choice. With the ability to produce valid seed messages and knowledge of the master key in the development kit's default setting, it was possible to pair it with the receiver by the *The secure learn procedure* and then completely emulate functionality of a physical transmitter.

6.1.1 Plaintext part

The plaintext (unencrypted) part of the KeeLoq message consists of 2 fixed zero bits, 4 bits for button status, and 28 bits of a serial number. When the serial number is changed to a value that does not correspond to any previously learned value, the message is rejected. If the button status does not correspond to button status contained in the encrypted part of the message, the message is rejected as well.

From the experiments, it seems like both the button status and the discrimination value in the encrypted part of the message are by default compared to values extracted from the plaintext part of the message to check the integrity of the decryption.

6.1.2 Ciphertext part

Besides the button status and discrimination value already covered in the previous subsection, the encrypted part also contains 2 overflow bits and 16 bits for the actual counter value. Experiments showed that the overflow bits are completely ignored by the receiver, which is supported by [22].

Discrimination value is usually equal to the lowest 10 bits of the serial number. The discrimination value based on the serial number allows the learning procedure to stay simple, as the value is directly contained in the regular message. KeeLoq system specification allows manufacturers to choose an arbitrary number as the discrimination value. However, as the number has only 10 bits, which means 2^{10} or 1024 possible values, choosing manufacturer-specific derivation of the discrimination value does not provide any substantial improvement of the overall security of the cryptosystem, because when the device key is disclosed, it can be easily obtained from any eavesdropped message.

6.2 Bruteforcing the key derivation

In 2009 [23] presented a brute-force attack with special FPGA-based cracking hardware COPACOBANA with the capability of massive parallelization of the KeeLoq decryption operation. The authors used pipelining and multiple units running in parallel to maximize the cracking power of the device.

The attack is focusing on the recovery of the device key based on only two captured messages from the transmitter, which can be easily obtained by passive sniffing of wireless communication. With knowledge of the manufacturer key and the knowledge of the key derivation algorithm, which consists of decryptions of the 64-bit seed divided into two 32-bit halves, the device key may be found by an exhaustive search among possible seed values. Procedures with the ability to obtain the manufacturer key are more profoundly described in 6.6. The entropy of the seed is determined by the bit length of the used seed word, so it limits the guessing space to either 2^{32} , 2^{48} , or 2^{60} possible seed values.

The paper shows that 32-bit and 48-bit seeds cannot be considered as secure. One instance of COPACOBANA cracking hardware is capable of revealing the device key derived from 32-bit seed in less than 1 second and device key derived from 48-bit seed in less than 6 hours. The 60-bit seed authors consider as secure enough, as the one instance of COPACOBANA would need 1011 days to crack the device key based on this seed. By using 100 instances of COPACOBANA, the time can be reduced to approximately 10 days. However, the tremendous cost of the hardware limits the practical use of the attack for the strongest possible 60-bit seed-based device key, which is also the one used by the reference implementation in the development kit.

The version of COPACOBANA used for the attack described in the paper, according to one of its authors, Martin Novotný, consisted of 120 Xilinx

Spartan3-1000 FPGA boards. The cost of such hardware makes it less available for a general audience.

However, newer research [24] presented improved attack using only one high-end FPGA from Xilinx Virtex family. Their improvements of the underlying decomposition of the cipher and using particular features of the Virtex-6 FPGA makes this attack more than 2 times faster than COPACOBANA. Even with cheaper Virtex-4 FPGA, the time needed to crack the device key derived from 48-bit seed is about 17 hours. These results make this attack applicable for 32-bit and 48-bit seed-derived keys, even without expensive hardware resources.

6.3 Bruteforcing the encrypted part

Papers about the security of a particular cryptosystem usually focus on bruteforcing the cryptographic algorithm itself to recover the key. There are 2^{32} possible keywords in the KeeLoq cryptosystem. Regarding the fact that KeeLoq accepts codes in a window of 16, it reduces the complexity of brute-force attempts to guess the correct encrypted message to 2^{28} , calculated as $2^{32}/16$. The probability of guessing two consecutive numbers in the resynchronization window is negligible.

For example, [16] calculates with a theoretical value of 10 guesses per second, which means one transmission every 100 ms. However, an experiment made during work on this thesis shows that minimal required time for transmission correctly received by the receiver side on the official development kit cannot be squeezed under 150 ms when using the default ASK-OOK modulation at 433.92 MHz. This attack, with approximately 466 days ($150 \text{ ms} * 2^{28}$) for the worst-case scenario, is useless in practice.

6.4 Desynchronizing the counter

The intention of an attacker does not have to be only gain of access to an asset protected by a remote keyless entry system. The attacker may want to prevent legitimate users from regularly using the system. There are several ways of how a denial-of-service can be provided.

Except a simple RF signal jamming (already covered in Subsection 1.3.1), another way to make a DoS attack to the KeeLoq RKE system is to desynchronize the counter value stored for a particular transmitter in the receiver. In order to change the counter value to a substantially higher number, two consecutive messages with counter values close to the upper bound of the resynchronization window must be transmitted. This could make the transmitter non-functioning for up to the next 32768 (2^{15}) key presses when the resynchronization is performed multiple times to value close to the end of resynchronization window.

Nevertheless, to transmit two consecutive messages within the resynchronization window, the attacker either needs to know the device key of the transmitter, or replay previously eavesdropped messages.

Possible ways to recover the device key are mostly described in Chapter 5 and Section 6.6. With the knowledge of the device key, the attacker can easily forge the necessary message in software and send them (for example, via SDR) to the receiver.

Two things need to be taken under consideration regarding the possibility of the replay of previously eavesdropped messages. As the experiments stated in the first subsection of this chapter, the receiver may ignore the overflow bits contained in the encrypted parts of the message, so when the counter overflows, messages from the previous cycle of the counter can be possibly replayed.

However, when we use the KeeLoq assumption [9] of an average of 10 operations per day, first possible moment, when the replayed message would be considered as a valid message would happen approximately after 9 years (precisely $(2^{15} + 1)/(365 * 10)$ years), which makes this variant of DoS attack practically unusable under these theoretical assumptions.

6.5 Breaking the counter

The newly presented attack is covered separately in Chapter 7.

6.6 Side-channel analysis

6.6.1 Differential power analysis

The amount of work related to side-channel power analysis is growing since its introduction to the cryptanalytic community. Until the release of a paper, which showed practical differential power analysis (DPA) attacks on standard KeeLoq implementations [22], side-channel attacks were mostly described to already vulnerable platforms or for unrealistic setups, like with artificial trigger signals for trace alignment. Researchers were able to produce practical attacks on black-box hardware, just with the knowledge of cipher properties.

The research shows that typical transmitters from Microchip line HC-Sxxx are equipped with hardware implementation of KeeLoq encryption, while the receivers based on PIC microcontrollers use software implementation of KeeLoq decryption.

As the transmitter in KeeLoq hopping code mode of operation doesn't provide plaintext of encrypted data, the attack flow is backward from the last encryption round, as the value of the last round result is known, as it's equal to the ciphertext part of the message, which can be easily eavesdropped. For the attack itself, researchers used DPA, based on the Hamming distance

power consumption model, which models leakages of the state register and extracted peaks from traces. With the use of an oscilloscope with a sampling rate of 200 MS/s and just 30 power traces, measured by hooking the probe to a simple shunt resistor, it's possible to, by described algorithm, recover any key for any transmitter from line HCSxxx. The whole attack (including trace measurements, preprocessing of data, and the algorithm itself) can be performed within one hour.

The second described attack focuses on the software implementation in receivers. As the authors stated, observed properties showed that decryption implementation flow is probably data-dependent, so the attack with a model based on Hamming weight needed about 10000 power traces to recover the full 64 bits of the manufacturer key using a similar algorithm as for hardware implementation. Even though this attack is not that convenient as the one on hardware implementation, it allows to exploit the manufacturer key, which is usually shared across large amounts of receivers from the same manufacturer, thus performing this attack once should be enough, when the attacker concentrates on targets from particular manufacturing series.

When the attacker has access to the manufacturer key, it can exploit the whole ecosystem using the key. For example, when the serial-based key derivation is used, it's possible to clone any transmitter by eavesdropping just one message, as the device key is easily recoverable (by known key derivation for the serial number-based scheme) from plaintext part of the message. The current counter value can also be recovered by decrypting the ciphertext part of the message with the derived key. The knowledge of the manufacturer key can be used even for recovering device keys based on some seed-based key derivation schemes, as shows the attack described in 6.2.

6.6.2 Simple power analysis

As authors of [22] (mentioned in the previous section) suggested, the software implementation may include data-dependent operation flow. The following work [25] proved that assumption right. This paper, describing the possibilities of simple power analysis (SPA) for software implementation shows, that typical implementation of KeeLoq decryption done in software, contains almost textbook SPA vulnerability. The attack was provided with an oscilloscope hooked to shunt resistor, with a sampling rate 125 MS/s.

By disassembling the code and visually analyzing the traces after peak extraction, authors discovered that the number of peaks in each decryption round modulo 2 equals the value of corresponding key bits. Such weakness allows exploiting the manufacturer key just with access to one power trace. This discovery may have devastating consequences for systems based on KeeLoq (as described in 6.6.1), as this flaw is contained in several distinct KeeLoq receivers, that authors tested.

New attack on KeeLoq counter

7.1 Introduction

Deep research of the KeeLoq RKE system and the complex viewpoint on the implementation of the KeeLoq system gained throughout the work on this thesis resulted in an attack, which enables the exploitation of the whole system in about an hour, in a practically undetectable way by a regular legitimate user.

This attack is unique by requiring zero knowledge about the used encryption algorithm and its properties, which makes it distinct from all other described attacks in this thesis based on related research papers. It also requires only limited hardware resources, as it may be executed from regular PC with connected SDR and a device to perform button presses on the transmitter.

7.2 Attack scope

The main limitation of this attack is the necessity of having physical access to the transmitting keyfob, but only for a limited time. However, even this requirement doesn't make the attack completely unfeasible, as side-channel attacks work with the same prerequisite.

Approximately one hour is a relatively short time, in which the legitimate user can leave the keyfob unattended or until the user realizes, for example, in case of the lost keyfob, that it's missing. The same thing applies to car rental companies as the key is in possession of a potential attacker for much longer than needed for the attack itself.

7.3 Exploitable properties of the system

The limited bit length of the counter, which is used to prevent a replay attack, is the main weakness, making this attack applicable to exploit the KeeLoq

RKE system. The counter has only 16 bits, which equals to a period of 65536 values. With the manufacturer assumption [26] of an average of 10 operations (i.e. lock, unlock, ...) per day, means that the counter should overflow after approximately 18 years (see calculation in 6.4), which is satisfactory regarding the typical lifespan of an asset (like a car) protected by KeeLoq RKE system. However, when this assumption is broken, and messages are sent in the maximum possible rate, it squeezes the required time to perform the attack to approximately one hour, as stated at the beginning of this section (6.5) and precisely calculated in 7.5.

As stated in 6.1.2, the receiver in the reference implementation, provided in the development kit (which contains MCS3142 transmitter/keyfob), completely ignores two counter overflow bits. This observation supports [22], which states that only two Microchip KeeLoq RKE products, which belong to product line HCS2xx, implement this feature correctly. Other products, like from line HCS3xx, don't implement this feature according to the authors' findings.

According to the KeeLoq documentation [26], when the overflow bits are set to 1 in transmitter's memory during the manufacturing process, the first bit is cleared (set to 0), when its counter overflows for the first time, which happens after 2^{16} transmitted messages. The second bit is cleared when the counter overflows for the second time. Other overflows don't modify the overflow bits, as the clearing of the bits is permanent. When implementation ignores the overflow bits, or the bits are already set to 0 during the manufacturing process, it makes all messages transmitted 2^{16} apart from one another equivalent, which makes the attack a few times faster, than in case of implementation according to the specifications using overflow bits.

7.4 Attack description

7.4.1 Capture part

Once the transmitter is hooked up to the hardware attack setup, the actual attack may start. It is supposed that the transmitter is out of the range of the receiver. In the beginning, the SDR is set to record the incoming signals at the transmitter's operating frequency, which is in case of the default configuration of the KeeLoq development kit set up to 433.92 MHz. Once the recording is started, one button (i.e., corresponding to close function) is pressed twice by the press simulator. The double operation is done for future resynchronization capability. Other buttons need to be pressed just once after these initial two presses. The maximum number of connected buttons to the KeeLoq transmitter is 4, which means that only up to 5 messages (let's call this number K) need to be captured at this phase. After the capture of these K presses, the SDR recording is stopped.

The press simulator is then set to dispatch $(2^{15} - K)$ messages. The SDR is then started to record once again. One button is pressed 2 times, followed by single presses of each other buttons, which results in K captured messages again. The SDR recording is stopped afterward as well. The double messages are then used for resynchronization of the timer.

If the attacker can replay some messages to the receiver before the legitimate user of the transmitter tries to use the transmitter used in the capture part, no more actions with the transmitter are necessary, and the capture part may end at this moment. Otherwise, the attacker needs to dispatch other $(2^{15} - K)$ messages, which sets the transmitters counter to the original value, as it increased by 2^{16} , which overflows the counter. This additional step would allow the legitimate user to use the transmitter as before with no obstructions. In this case, the transmitter is no longer needed.

7.4.2 Replay part

When the attacker executed the attack with $(2^{15} + K)$ emulated presses and can replay some messages before the legitimate user uses the transmitter (one used in the capture part), the attacker replays first pair (double operation) from the first capture, followed by the first pair of the messages from the second capture. This procedure resynchronizes the receiver's counter to value a few presses below $(\Theta + 2^{15} + 2)$ the current counter value of the transmitter $(\Theta + 2^{15} + K)$, where Θ is the counter value at the beginning of the attack. This traversal allows the legitimate user to use the transmitter right away, as the next number contained in the message by the transmitter will most likely be in a valid window of 16 messages, relative to the resynchronized value.

The two captured pairs (double operation) apart 2^{15} messages from each other allows the attacker to traverse between both halves of the counter period (see Figure 7.2). When one double operation is replayed, the relative resynchronization window includes double operation from the other half of the period, as shows dashed line in Figure 7.2. This capability applies for both pairs (2^{15} values apart), which allows the attacker to be always able to transmit messages valid for the receiver at any time using one traversal at maximum. It also allows the attacker to return back to the appropriate half of the period, where the legitimate user's counter is synchronized after the second press as it falls to the corresponding window for the counter resynchronization. This feature makes the manipulation of the counter by the cautious attacker basically undetectable for legitimate users.

Let's assume the following scenario. An attacker successfully executed the capture part on the car keyfob with the resynchronizing replay. After some time it is almost certain, that the current value is somewhere between $(\Theta + 2^{15} + K)$ and Θ (see 7.4.2). When the attacker wants to break into the car, he needs to replay the first double operation (i.e., close) and then replay open operation, which was captured in the capture phase right after

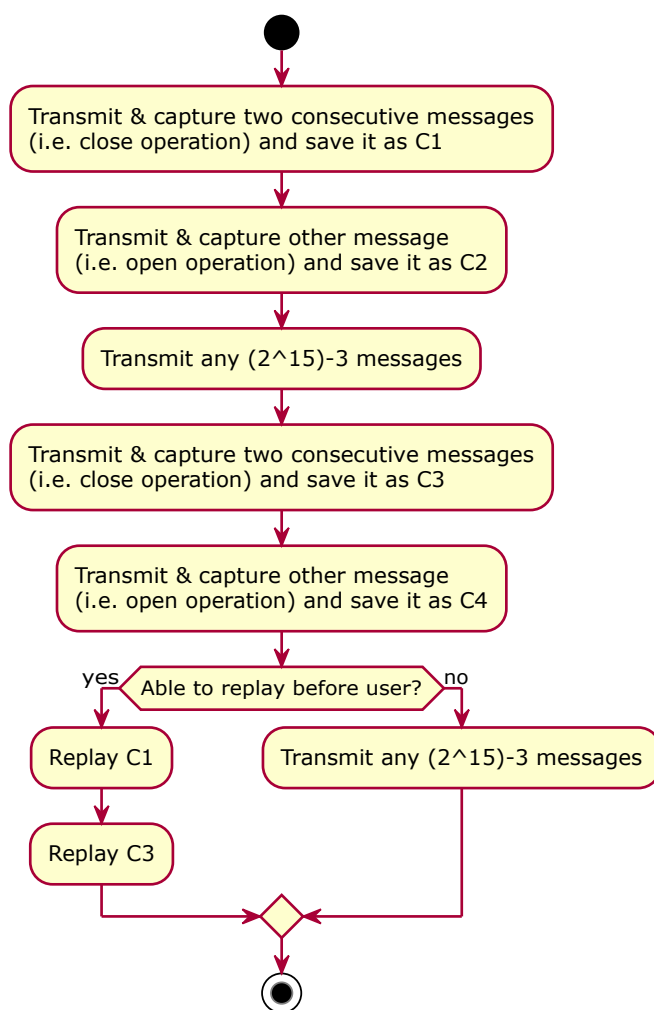


Figure 7.1: Attack flow for implementations ignoring overflow bits

the first double operation. Once the attacker burglarizes the car, he replays the second double operation from the second half of the counter period (starting at $\Theta + 2^{15}$), and the attack is over. When later legitimate user tries to open the car, the first message fails to open the car, as the receiver is waiting on the second consecutive message, which would probably be sent by the user right after and would not raise any suspicion at that time.

7.5 Attack duration

To make transmission stable, regarding the logic timing of the transmitter, the message transmissions (or more precisely simulated keypresses) need to

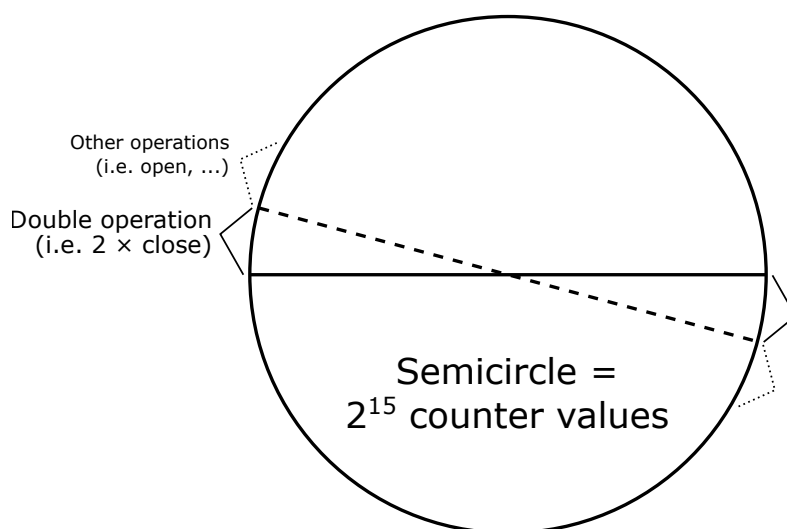


Figure 7.2: Illustration for better understanding of traversal between both halves of the counter period

be apart from one another in some arbitrary time intervals. The author's experiments showed that the minimum time interval of emulated key presses is 115 ms. It means, that in the ideal case, when the attacker is not limited by the inability to replay some messages before the legitimate user, the whole attack would take only slightly more than one hour, as shows equation 7.1, which assumes $K = 3$ (2 distinct buttons to record, i.e., close and open).

$$t = 115 \text{ ms} \times (2^{15} + 3) = 115 \text{ ms} \times 32771 = 3768665 \text{ ms} < 63 \text{ minutes} \quad (7.1)$$

Let's calculate with 63 minutes as the time to go through one half of the counter period. The time needed for the case, when the attacker is not able to replay before the legitimate user (right branch in Figure 7.1), would be approximately doubled, meaning total time below 2 hours and 6 minutes.

In case that the attack is performed on implementation, which uses overflow bits properly, it would be necessary to execute the longer flow of the attack (right branch in Figure 7.1) 2 times, as the current counter value would most likely be not overflowed. Later it would be needed to replay C1 and C3 double operations in captured order for both flows before the described procedure may be applied. This would increase the time calculated previously by approximately 4 hours and 12 minutes $((2 \times 2 \times 63) = 252 \text{ minutes})$.

7.6 Countermeasures

This vulnerability is deeply embedded in the KeeLoq system's design, so there are not many countermeasures that developers utilizing the KeeLoq system

can implement. One possibility is to take into consideration overflow bits in a way that extends the effective period of the counter twice (when the first overflow bit is set). In this case, the receiver would reject messages coming from the transmitter having both overflow bits cleared, meaning two overflows occurred. It would allow mitigating this attack, and it wouldn't interfere with regular use of the transmitter, as with manufacturer assumptions [26] it would last for approximately 36 years (2×18 years), which is most likely longer than the lifespan of the system. But in case of this countermeasure being implemented, if someone would try to apply this attack, the keyfob would become unusable and would need to be replaced. In most cases, the cost of the keyfob is much lower than the loss caused by a criminal opening a car, garage, or house in a practically undetectable way, causing further damage by issues with insurance policies.

7.7 Future work

The following work should test this attack on some specific real KeeLoq applications, where the demonstration platform presented in Chapter 8 may help with the executions of the attack.

Also, the button press simulator now works on a purely electronic base (transistor switching the circuit, see Figure 8.3), where the access to PCB of transmitter is needed. So, it would be nice to have a device that simulates the button presses on a mechanical basis, limiting the need for invasive actions on the transmitter.

Demonstration platform

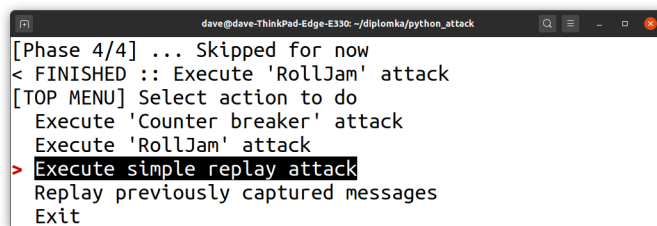
8.1 Common hardware and software setup

The hardware setup consists of SDR USRP B210, with two connected antennas, one for the transmission and another for the reception. It also contains the KeeLoq DM182017-4 development kit, which contains display for visual feedback of the last processed event in the receiver. One of the transmitting keyfobs is used for demonstration for replay and RollJam attacks, and the other one is dedicated to the counter breaking attack.



Figure 8.1: Common hardware setup of the demonstration platform

Software is written in the Python 3 language, and it depends on several libraries, like UHD (for SDR), NumPy, threading, and few others. Development and testing were done on a Linux machine running Ubuntu 20.04. The user interface is terminal-based and uses *Simple Terminal Menu* library for menu-like selectors, which are used several times, when user interaction is needed. All the libraries are freely available through system repositories or Python package index PyPI. The software is initiated by executing the main script from the terminal.



```
dave@dave-ThinkPad-Edge-E330: ~/diplomka/python_attack
[Phase 4/4] ... Skipped for now
< FINISHED :: Execute 'RollJam' attack
[TOP MENU] Select action to do
Execute 'Counter breaker' attack
Execute 'RollJam' attack
> Execute simple replay attack
Replay previously captured messages
Exit
```

Figure 8.2: Terminal-based user interface

The software is written as a multi-threaded application. The main thread process the UI events, prints logs, saves and loads captured data, and manages SDR operations. In the case of Counter breaking attack implementation (see 8.5), it also takes care of sending commands to the button press simulator and timing between presses.

The SDR operations are limited to receive and transmit signals for a specified time. They start to run from a separate worker thread, which is initiated from the main thread. It's possible to stop the running operation, before a specified timeout, from the main thread. When the specified action timeouts or is stopped early, the worker thread terminates and then is joined by the main thread.

8.2 Features

The software support attacks described in the following sections. All attacks are initiated by choosing a particular option from the main menu. Some of the captures made during attack flows are accessible to replay via another option in the main menu.

8.3 Replay attack

The replay attack is straightforward. The program receives a signal for 5 seconds and then prompts the user if the replay should be done right away. If not, it's possible to replay the received capture later from the main menu by the replay choice.

8.4 RollJam attack

RollJam attack implementation is based on the flow described in Figure 1.5. For better orientation, the attack is divided into 4 phases. The user is always prompted to make a transition to the next phase.

In phase 1, the first message is captured. In phase 2, the second message is captured. Immediately after phase 2, at the time, when jamming should stop, phase 3 starts, and the first message is replayed. Phase 4 is optional, and if the user doesn't want to replay the second message right away, the attack ends. It's possible to replay the second message going to the replay section accessible from the main menu.

8.5 Counter breaking attack

8.5.1 Specific setup

By having full access to the development kit hardware, the hardware attack setup is provided by the second transmitting keyfob, which is decapped from its plastic housing, and button pads are attached to a microcontroller's (ESP8266) GPIO pins. This MCU was chosen for support of the MicroPython environment, which makes quick prototyping development easy, thanks to high-level APIs. It's widely available and has adequate support for GPIO manipulations necessary for this attack.

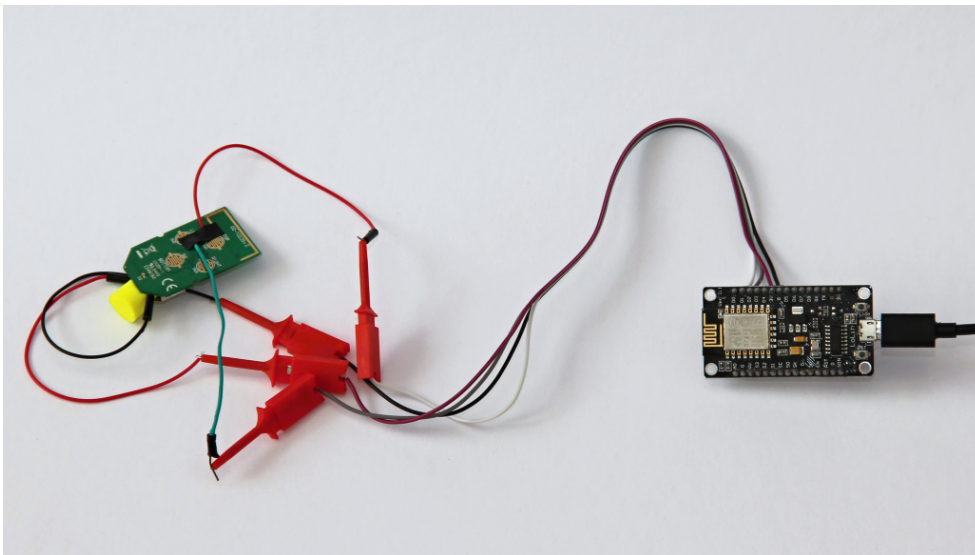


Figure 8.3: Button press simulator hardware

The button pad of the transmitter is under normal circumstances connected to the transmitter's ground when a particular physical button is pressed.

GPIO pins connected to button pads are set up as open-drain outputs, which connect the pin to the ground by activating an inner transistor assigned to this particular GPIO unit. This setup makes connecting required button pads to the ground possible through the microcontroller’s software, which emulates button press actions.

The microcontroller software setup is relatively simple, as it only needs to emulate button presses, by closing a circuit via GPIO for 20 ms, which is enough for transmitter’s pin debouncing procedure. Computer software sends commands to ESP8266 over the built-in serial (UART) connection. The command describes which one of two buttons should be “pressed”.

8.5.2 Description

Counter breaking attack, the new attack described in this thesis, follows the flow described in Figure 7.1. It is also for better orientation divided into 4 phases. All phases, except the replay, are possible and recommended to perform outside the receiver range.

In phase 1, the SDR records double operation, which is initiated by two messages asking to emulate press of button 2 (\sim close), and single operation, also with the corresponding command to emulate press of button 1 (\sim open). These two captures are saved. In phase 2, the SDR is idle, and software executes $(2^{15} - 3)$ emulated key presses. It could be either one button and implementation uses button 2. This phase takes more than one hour to complete. Phase 3 is equivalent to phase 1. After phase 3, the user is prompted if he’s able to replay at the moment. If so, phase 4 consists of replaying double messages from phases 1 and 3. The replay may be skipped if it can be executed later, so phase 4 is skipped in this case. If the replay cannot be sent to the receiver before a legitimate user uses the transmitter, software execute $(2^{15} - 3)$ emulated key presses in phase 4, while the SDR is idle. This additional step effectively resets the transmitter’s counter to its original value by overflowing the counter, as 2^{16} presses are executed in total.

Captures of a double and a single operation may be replayed from the replay section in the main menu. By having the captures saved persistently, it’s possible to execute any operation, either corresponding to button 1 or button 2 anytime later. It may be necessary to traverse between the counter period halves and return in the same manner to the proper half of the counter in order not to restrict legitimate user’s use of the system.

Conclusion

The objectives of this thesis were to analyze commonly used RKE platforms (Chapters 1 and 4), proceed with in-depth security analysis with increased focus on implementation weaknesses (Chapters 5, 6 and 7) and implement some selected attacks using SDR (Chapter 8). The author is convinced, that all these objectives have been met.

With the perspective of all information gathered during the work on this thesis, the ideal RKE system should have the following properties. It should be an active RKE system to prevent relay attacks, which are especially potent in the case of passive keyless entry systems. It should use a provably secure encryption algorithm (like AES); as the HITAG2 case showed, it may be only a matter of time until proprietary cipher becomes broken. Also, it's crucial to implement a robust and secure key distribution scheme because even without an applicable cryptanalytic attack, the security of the system may be compromised by weak key management, like with some variants of RKE systems based on AUT64 or KeeLoq. It should use non-repeating nonces, like RTC timestamps, which can also prevent replay-based attacks, like Roll-Jam. Finally, both HW and SW should be designed with side-channel attacks countermeasures in mind.

This thesis also presents a new attack (in Chapter 7), which allows exploiting the KeeLoq RKE system, when an attacker has physical access to a transmitter. Unlike in the mentioned DPA attack case, this attack may be performed without costly hardware equipment (like an oscilloscope), as it uses only an SDR and a device simulating button presses. This attack exploits weaknesses in the system design of the rolling code counter and another flaw in the system's implementation. In a practical described scenario, this attack is feasible to perform in time of approximately one hour. The author believes that with some proposed improvements, this attack has the potential to be published as a standalone article.

Bibliography

- [1] Garcia, F. D.; Oswald, D.; et al. Lock it and still lose it—on the (in) security of automotive remote keyless entry systems. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016.
- [2] Microchip Technology Inc. *MCS3142 Dual KEELOQ Encoder Wireless Remote Control Development Kit User's Guide [online]*. 2014, [Cited 2020-05-20]. Available from: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001746A.pdf>
- [3] Francillon, A.; Danev, B.; et al. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011.
- [4] Pardhasaradhi, A.; Kumar, R. R. Signal Jamming and Its Modern Applications. *International Journal of Science and Research*, volume 2.
- [5] Kamkar, S. Drive it like you hacked it: New attacks and tools to wirelessly steal cars. *Presentation at DEFCON*, volume 23, 2015.
- [6] Modulation and multiplexing [online]. <https://notes.eddyerburgh.me/computer-networking/modulation-and-multiplexing#baseband-transmission>, [Cited 2020-05-25].
- [7] What is WiFi 6? (802.11 ax) – TP Link [online]. <https://www.tp-link.com/us/wifi6>, 2020, [Cited 2020-05-25].
- [8] Line code – Wikipedia, The Free Encyclopedia [online]. https://en.wikipedia.org/w/index.php?title=Line_code&oldid=953268976, 2020, [Cited 2020-05-25].
- [9] Microchip Technology Inc. *MCS3142 Dual KEELOQ Technology Encoder Data Sheet [online]*. 2014, [Cited 2020-05-20]. Available from: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001746A.pdf>

- [10] LuaRadio – New to SDR? [online]. <https://luaradio.io/new-to-sdr.html>, [Cited 2020-05-25].
- [11] Ettus Research. *USRP B200/B210 Bus Series [online]*. 2019, [Cited 2020-05-25]. Available from: https://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf
- [12] Hicks, C.; Garcia, F. D.; et al. Dismantling the AUT64 Automotive Cipher. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018: pp. 46–69.
- [13] Eli Biham, S. I. N. K., Orr Dunkelman; Preneel, B. How To Steal Cars – A Practical Attack on KeeLoq [online]. <https://www.cosic.esat.kuleuven.be/keeloq>, [Cited 2020-05-25].
- [14] Courtois, N. T.; O’Neil, S.; et al. Practical algebraic attacks on the Hitag2 stream cipher. In *International Conference on Information Security*, Springer, 2009, pp. 167–176.
- [15] Verstegen, A.; Verdult, R.; et al. Hitag 2 Hell – Brutally Optimizing Guess-and-Determine Attacks. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD: USENIX Association, Aug. 2018. Available from: <https://www.usenix.org/conference/woot18/presentation/verstegen>
- [16] Enderlein, R. R. KeeLoq. 2010. Available from: <http://www.e7n.ch/data/e10.pdf>
- [17] Microchip Technology Inc. *Secure Learning RKE Systems Using KEELOQ Encoders [online]*. 1996, [Cited 2020-05-25]. Available from: <http://ww1.microchip.com/downloads/en/AppNotes/91000a.pdf>
- [18] Courtois, N. T.; Bard, G. V.; et al. Algebraic and slide attacks on KeeLoq. In *International Workshop on Fast Software Encryption*, Springer, 2008, pp. 97–115.
- [19] Indestege, S.; Keller, N.; et al. A practical attack on KeeLoq. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2008, pp. 1–18.
- [20] Bogdanov, A. Cryptanalysis of the KeeLoq block cipher. *IACR Cryptology ePrint Archive*, volume 2007, 2007: p. 55.
- [21] Eisenbarth, T.; Kasper, T.; et al. *Keeloq*. Boston, MA: Springer US, 2011, ISBN 978-1-4419-5906-5, pp. 671–673, doi:10.1007/978-1-4419-5906-5_587. Available from: https://doi.org/10.1007/978-1-4419-5906-5_587

- [22] Eisenbarth, T.; Kasper, T.; et al. Physical Cryptanalysis of KeeLoq Code Hopping Applications. *IACR Cryptology ePrint Archive*, volume 2008, 2008: p. 58.
- [23] Novotný, M.; Kasper, T. Cryptanalysis of KeeLoq with COPACOBANA. In *Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS 2009)*, 2009, pp. 159–164.
- [24] Sheetrit, I.; Wool, A. *Cryptanalysis of KeeLoq code-hopping using a Single FPGA*. Tel Aviv University, 2011.
- [25] Kasper, M.; Kasper, T.; et al. Breaking KeeLoq in a flash: on extracting keys at lightning speed. In *International Conference on Cryptology in Africa*, Springer, 2009, pp. 403–420.
- [26] Microchip Technology Inc. *HCS301 KEELOQ Code Hopping Encoder [online]*. 2001, [Cited 2020-05-25]. Available from: <http://ww1.microchip.com/downloads/en/devicedoc/21143b.pdf>

Acronyms

API	Application programming interface
ASK	Amplitude-shift keying
DoS	Denial of Service
DPA	Differential power analysis
FPGA	Field-programmable gate array
FSK	Frequency-shift keying
GPIO	General purpose input output
GPU	Graphics processing unit
GUI	Graphical user interface
ISM	Industrial, scientific and medical (radio bands)
KP	Known-plaintext
LO	Local oscillator
MCU	Microcontroller unit
NLFSR	Nonlinear-feedback shift register
OOK	On-off keying
PCB	Printed circuit board
PSK	Phase-shift keying
RF	Radio-frequency
RKE	Remote keyless entry

A. ACRONYMS

RTC Real-time clock

SDR Software-defined radio

SNR Signal-to-noise ratio

SPA Simple power analysis

URH Universal radio hacker

XML eXtensible Markup Language

Contents of enclosed CD

	readme.txt	the file with CD contents description
	src	the directory of source codes
	code	the directory of demonstration platform implementation
	thesis	the directory of L ^A T _E X source codes of the thesis
	text	the thesis text directory
	thesis.pdf	the thesis text in PDF format